AFRL-IF-RS-TR-2005-256
Final Technical Report
July 2005

ACTIVELETS: LIGHT WEIGHT ARCHITECTURE FOR SPECIAL OPERATIONS FORCES (SOF) PLANNING AND SCHEDULING

Carnegie Mellon University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J737

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** JULY 2005	**3. REPORT TYPE AND DATES COVERED** Final Apr 00 – Dec 03

4. TITLE AND SUBTITLE ACTIVELETS: LIGHT WEIGHT ARCHITECTURE FOR SPECIAL OPERATIONS FORCES (SOF) PLANNING AND SCHEDULING	**5. FUNDING NUMBERS** C - F30602-00-2-0503 PE - 63760E PR - ATEM TA - P0 WU - 01
6. AUTHOR(S) Stephen F. Smith, David W. Hildum, and David R. Crimm	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Carnegie Mellon University 5000 Forbes Avenue Pittsburgh Pennsylvania 15213	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency 3701 North Fairfax Drive Arlington Virginia 22203-1714 — AFRL/IFSA 525 Brooks Road Rome New York 13441-4505	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2005-256

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: John F. Lemmer/IFSA/(315) 330-3657/ John.Lemmer@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(Maximum 200 Words)***

This project aimed at delivering continuous planning and scheduling capabilities to support crisis response for Special Operations Forces. The principal accomplishments of this effort has been the development of Comirem (COntinuous Mixed-Initiative REsource Management tool), an interactive planning/scheduling system designed to more directly match the iterative user-centered nature of planning and scheduling in practical domains by providing the means to represent evolving scenarios, define and reuse operational templates, maintain problem consistency in response to execution monitoring and information updates, and detect inconsistencies and signal appropriate alerts in response. This report describes the design of Comirem and summarizes the capabilities that it provides.

14. SUBJECT TERMS Continuous Planning, Mixed-Initiative Resource Management, Interactive Planning-Scheduling	**15. NUMBER OF PAGES** 147
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Contents

List of Figures

List of Tables

1 Overview

This document is the final report for DARPA contract # F30602-00-2-0503, entitled “Activelets: Light Weight Architectures for SOF Planning and Scheduling.” As part of the **Active Templates** program, this project aimed broadly at delivering continuous planning and scheduling capabilities to support crisis response for Special Operations Forces. The principal accomplishment of this effort has been the development of Comirem (COntinuous Mixed-Initiative REsource Management tool), an interactive planning/scheduling system designed to more directly match the iterative, user-centered nature of planning and scheduling in practical domains by providing the means to represent evolving scenarios, define and reuse operational templates, maintain problem consistency in response to execution monitoring and information updates, and detect inconsistencies and signal appropriate alerts in response. This report describes the design of Comirem and summarizes the capabilities that it provides.

A significant historical limitation of planning and scheduling technologies is that they tend to be designed as batch-oriented solution generators, where users cannot directly guide solution development and attempt to achieve desired results indirectly by manipulating system inputs. This user-interaction model and system design perspective is at direct odds with the characteristics of most practical domains, where requirements, capabilities and plans evolve incrementally and in parallel over time, and users invariably possess knowledge that should override aspects of system models. Furthermore, the need for reassessment and revision continues as the plan is executed and results deviate from expectations. In practice, planning and resource management is fundamentally an *iterative* process, and attempting to support this process with a batch-oriented solver results in an awkward, indirect and inherently inefficient problem-solving cycle.

Comirem is designed from an alternative, incremental perspective. It promotes a graphical, spreadsheet-like model of user-system interaction, wherein resource assignment decisions can be constrained, refined and revised opportunistically as information becomes available, as specific planning choices become apparent, as execution results dictate, or as circumstances otherwise warrant. The system provides a range of support for a user-driven resource management process. At any point in the planning process, the system computes and maintains a basic set of resource allocation options for any given task in the plan. As task requirements and resource capabilities are adjusted, or as specific resource assignments are made, underlying time and resource propagation techniques are applied to update the feasible options for other tasks, to identify forced decisions and to detect infeasible solutions. The user can invoke an automated scheduling capability to establish overall resource feasibility of a plan, and this capability can be coupled with an undo mechanism to explore the consequences of various requirement and/or resource availability changes. All interaction with the system is in terms of a high-level domain model, which shields the user from the details of the system’s underlying search model. Comirem has been applied initially to the domain of Special Operations Forces (SOF) planning, where the use of transportation assets, taskforces and equipment must be managed over time.

The Comirem effort descends from earlier work within the Intelligent Coordination and Logistics Laboratory in the Robotics Institute at Carnegie Mellon University involving the AMC Barrel Allocator [Becker and Smith, 2000], a tool for day-to-day allocation of aircraft and aircrews for the United States Air Force that is now part of the operational planning system in the Tanker/Airlift Control Center at the USAF Air Mobility Command. Whereas the AMC Barrel Allocator focuses on large-scale problems with more narrowly circumscribed types of allocation constraints, Comirem focuses on smaller-scale but more ill-structured types of allocation problems.

The design of the Comirem planner reflects several basic principles that we believe are fundamental to collaborative, human-computer interaction in practical planning domains:

- *Adjustable Decision-Making Autonomy*

  One key to effective collaborative planning is a decision-making process that allows the degree of automation to vary according to problem solving context and user preference. The Comirem planner allows the user to inject herself into the decision-making process at different levels of granularity. At the lowest level, the system takes responsibility only for proposing and implementing resource assignments for individual elements (or activities) in the plan, and the user retains control over resource assignment decisions. However, even in this (largely manual) mode of operation, the system is frequently playing a quite substantial role, in determining the consequences of all relevant constraints on user allocation decisions (e.g., the overall duration of a move, given that multiple trips are required and aircrews must rest during the course of the move). The user is free to make specific adjustments to an activity's resource requirements (e.g., adding another aircraft type to the set of possible allocation options) or to override specific usage constraints (e.g., reduce computed duration constraints due to knowledge of a strong tail wind) without carrying along and anticipating interactions with all other constraints. In other decision contexts the user can delegate increased decision-making scope to the system. The user may choose to manipulate constraints associated with larger plan fragments (e.g., plan threads), and request feasibility checking and/or scheduling in response to assess consequences. At the end of the automation continuum, the user can request the construction of a fully sourced plan, in which case the system constructs a schedule according to user-specified goals and preferences. Through iterative generation, retraction and adjustment of (sets of) resource assignments, system planning and scheduling processes support collaborative reconciliation of planning objectives and desired actions with available resources.

- *Translation of System Models and Decisions*

  A second key to effective collaborative planning is an ability on the part of the system to communicate elements of its internal models and solutions in user-comprehensible terms. In Comirem, this is accomplished through a combination of graphical visualization and model-based explanation techniques. Visual displays are used to compactly convey various decision implications (e.g., how many trips are required if this type of transportation resource is used, from where will various resources be sourced, what will be the overall makespan of a particular movement activity) as well as potentially non-obvious allocation constraints (e.g., MOG limitations). In other advice-giving contexts, internal system representations are mapped to user-level actions through the use of abstract domain models. Comirem's higher-level "activity-resource" model allows the system to impose structure on the distance constraints involved in a "cycle" that has been detected in the temporal constraint network (i.e., characterizing them as interdependent sets of activity durations, inter-activity precedence relations, etc.). This structure then provides the basis for identifying and proposing relevant, user-level constraint relaxation actions that make sense for resolving the conflict.

- *Incremental Problem-Solving Procedures*

  A third key to effective collaborative planning is a problem-solving process that promotes incremental decision making. In complex, ill-structured planning domains, all relevant constraints are generally not known at the outset, and those that are can frequently change as planning and execution progresses over time. Hence, planning proceeds with whatever information is available, decisions are made on the basis of current information, and as additional information becomes known, conflicting decisions are appropriately revised. Within such a continuous planning process, the ability to maintain a sense of problem-solving continuity to the user is crucial. In arriving at a given plan or schedule (either partial or complete), the user has invested time and energy in understanding, assessing and validating various component decisions. The introduction of new or changed constraints should not arbitrarily result in wholesale changes to the plan. The system should instead act to preserve prior decisions wherever possible and meaningful. The COMIREM planning framework is designed generally with an incremental, change-based problem-solving process in mind, and furthermore allows the user to explicitly manage the scope of allowable change in specific decision contexts.

The remainder of this report is organized as follows:

- Section 2 provides an analysis of the SOF planning domain, highlighting the issues that have shaped the development of COMIREM.
- Section 3 summarizes the basic components of a COMIREM domain model, including representational issues and the conceptual model, and describes the COMIREM scheduling engine and system architecture.
- Section 4 describes our efforts towards integration with other applications in the SOF-Tools environment.
- Section 5 presents two SOF-related demonstration problems that have driven both the design, implementation and evaluation of the COMIREM system.
- Finally, in Section 6, we discuss some potential extensions to the COMIREM system and scheduling model.
- Three appendices are also provided:
  - Appendix A presents a full description of the XML specification for all COMIREM input data files.
  - Appendix B includes a user's manual for the COMIREM web-browser-based GUI.
  - Appendix C presents the abstracts of all research publications funded under this project.

2 Domain Analysis

The current focus for Comirem is the Special Operations Forces (SOF) planning domain, which is characterized by a very fine-grained representation of activities and resources, a relatively short temporal horizon (typically a number of hours or days), and a highly synchronized environment where multiple mission threads must be coordinated as part of an overall mission (e.g., air support provided for the duration of an assault on a target). In addition, plans and schedules in the SOF domain must be developed rapidly on the basis of available (and often incomplete) information and then continually refined and revised as new information is accumulated and execution circumstances necessitate change. Execution typically commences before planning is complete and the dynamic nature of the environment requires a tight coupling between execution and (re)planning processes. Plans inevitably do not execute as expected, so there is a constant need to be able to quickly identify critical constraints and decision tradeoffs, and for rapid evaluation of alternative recovery options. The problem is further complicated by the fact that planning often takes place in a distributed fashion, by mobile decision-makers with limited computational resources and under executional duress.

Additional features related to the flexible nature of plans in the SOF domain are relevant to the design of Comirem:

- Activity requirements often include a choice of multiple resources or capabilities, and the selection of a particular resource or capability may impact decisions elsewhere in the plan by either restricting or relaxing remaining scheduling options.
- Resources may be aggregated and disaggregated over time as a means of grouping personnel and equipment into functional teams (e.g., *taskforces*).
- Resource constraints become known incrementally. As knowledge of the problem domain develops (e.g., the status, location and performance characteristics of resources), the system must be able to incorporate and adapt to changes in the underlying resource model.
- Resources may themselves be treated as cargo and transported by other resources, enabling the positioning of assets throughout the theater.
- Certain activities (e.g., those involving the sourcing of assets) may only be required in specific situations, whereupon their durations are dictated by the present circumstances. These *derivative* activities retain zero durations when not required.
- An activity's resource requirements may warrant multiple trips by a resource or resources to perform a task (e.g., the transportation of troops and equipment). *Sibling* activities (i.e., clones) are instantiated automatically on demand to represent situations where the cargo manifest for an activity exceeds the capacity limits of a single resource.
- Sequences of activities can be grouped into *threads* and assigned resources (e.g., taskforces) collectively to better reflect SOF mission-planning approaches.

3 COMIREM System Overview

COMIREM supports both incremental and automatic scheduling in logistical environments. In this section, we present an extensive description of the primary features and components of COMIREM that contribute to delivering its various scheduling capabilities, focusing on the conceptual model for representing logistical scheduling domains, the flexible-time-bounds scheduler and mixed-initiative techniques for resource management. The overall design of COMIREM reflects the following themes:

- *Flexible user control of scheduling automation:*

  The system supports both incremental (i.e., user-driven) and automatic (i.e., system-driven) commitment/decommitment of resources. Allocation options are continuously updated by the system to reflect the current state of decision-making and provide an up-to-date assessment of resource availability, and an undo mechanism supports a *what-if* style of interaction that facilitates consideration of alternative task-allocation decisions.

- *Constraint and solution visualization:*

  To provide further support for user interaction, the COMIREM GUI highlights the propagated impact of constraint tightening and relaxation (e.g., owing to the manipulation of temporal bounds) and changes in resource availability (e.g., owing to the commitment/decommitment of resources). A spreadsheet display metaphor is utilized throughout to provide direct access to relevant problem constraints and visualization of the impact of their modification.

- *Tight interplay between scheduling and planning decisions:*

  The distinction between what is generally thought of as a "scheduling" decision as opposed to a "planning" decision is becoming increasingly transparent. Planning decisions, like *how to perform a task* (i.e., the sequence of activities required to achieve its implied objective), are directly influenced by the typical scheduling decisions of *when to perform a task* and *which resource(s) to use*, and vice versa. COMIREM relies on an explicit and declarative domain knowledge base to make all relevant information available to the user and the system throughout the entire problem-solving process.

- *Lightweight component capabilities:*

  In a nod to the more lightweight client/server model, COMIREM is implemented as a standalone scheduling-engine server (written in Common Lisp [Steele Jr., 1990]) that interacts with an independent web-browser-based GUI client to deliver incremental and automatic scheduling capabilities to the user and set the stage for possible future multi-user interaction.

- *Interoperability:*

  The use of XML [Harold and Means, 2002] as the input/output data specification format facilitates interoperability with a wide range of external/legacy systems and provides for an open and explicit specification of all system inputs and outputs (i.e., at the level of both domain definition and problem description).

3.1 Representation of SOF Domains

Comirem allows the user to define a problem-solving context (called a *situation*) that is comprised of descriptions of domain-specific activities, resources and capabilities. These descriptions are built upon a set of system-provided situation-independent primitives, such as basic *move* and *event* activities, *stationary* and *mobile* resources, and temporal sequencing constraints. (These primitives are formally introduced in the following section, 3.2: **Conceptual Model**.) Situation descriptions are augmented by loading in one or more plan descriptions (called *scenarios*) defining networks of activities to which available resources must be assigned. Scenarios rely on the activity types, resource types and instances, and capabilities defined within the currently loaded situation. This separation of data allows the user to establish a specific context within which an evolving set of activities is to be scheduled.[1]

For example, a typical military situation may involve resource definitions for some basic helicopter types (e.g., `MH-47`, `MH-60`) and some instances of those types (e.g., tail numbers `MH-47-001`, `MH-47-001`, `MH-60-001`) initially located at a predefined location (e.g., `BASE-ALPHA`). Each helicopter type provides a set of capabilities (e.g., `heavy-transport`, `light-transport`) that may in turn be required by typical military activities (e.g., `deploy-forces`, `evacuate-forces`). These activity types may further include detailed descriptions of required subactions, such as cargo loading/unloading steps and positioning/depositioning legs that may be required of their assigned resources.

Given the situation described above, the stage is now set to define a rescue plan involving instances of the `deploy-forces` and `evacuate-forces` activity types that relies on the three helicopters available at `BASE-ALPHA` to deploy forces and perform a rescue and evacuation mission to/from some designated location (e.g., `embassy`). After loading both the situation and scenario (the latter of which identifies the necessary plan-specific deployment/rescue locations such as `embassy`), Comirem can be used to determine the feasibility of the plan given its various requirements and constraints and the available resources, keeping in mind that additional plans may have already been loaded and scheduled within this same situation.

A Comirem situation description includes the following pieces of information:

- *activity type definitions:* descriptions (i.e., attributes and their default values) of activity classes available for constructing a plan (e.g., `move-cargo`, `airdrop-forces`, `fighter-escort`)

- *resource type definitions:* descriptions (i.e., attributes and their default values) of all available resource classes (e.g., `F-16`, `B-2`, `MH-60`)

- *model type definitions:* hierarchical classifications of resource types relevant to a particular domain (e.g., `ROTOR` and `FIXED-WING` for aircraft, `WHEELED` and `TRACKED` for vehicles) that encapsulate common constraints and capabilities

[1] Only a single situation may be in effect (i.e., loaded) at any point in time; loading a situation erases any existing information about activity types, resources types and instances, and capabilities. Scenarios, on the other hand, are cumulative: that is, multiple plans can be loaded into a single situation to represent multiple demands (i.e., sets of activities) to be scheduled. (The exception is that once loaded, plan files cannot be reloaded into a situation.)

- *capabilities:* descriptions of capabilities (e.g., `heavy-transport`, `light-transport`) supported by resources and required by activities (to provide high-level links between activities and resources)

- *resource instances:* descriptions (i.e., sets of attribute/value pairs) of *all* resources available in a particular situation (e.g., `MH-47-001`, `MH-47-002`)[2]

Once a situation has been loaded into Comirem, any number of scenarios may be loaded and then scheduled using the various type and instance definitions made available by the situation. A scenario description includes the following pieces of information:

- *plan configuration:* description of a plan (e.g., its temporal horizon and granularity, release and due dates, etc.)

- *activity instances:* descriptions (i.e., sets of attribute/value pairs) of the activities that comprise a plan

- *constraint instances:* descriptions (i.e., sets of attribute/value pairs) of the constraints governing the temporal sequencing of activities in a plan[3]

- *place instances:* descriptions (i.e., sets of attribute/value pairs) of the places specific to a plan (e.g., locations referenced by the plan's activities)

Figure 1 identifies the grouping of problem structure and data information into situation and scenario files and specifies the order in which these files are loaded into Comirem.

3.2 Conceptual Model

The underlying scheduling domain model in Comirem is based on the Ozone *scheduling domain ontology* [Smith and Becker, 1997], which allows a user-interpretable description of an application domain to be mapped to application system functionalities. Problems are expressed in terms of *activities*, *resources* and *constraints*. An activity requires the use of one or more resources for some period of time to perform a designated task. Both activities and resources are governed by a set of constraints, which impose restrictions on when activities can execute, what resources can be assigned, and when resources are available for assignment. In the following subsections, we present these basic representational entities in further detail.

Additionally, Comirem supports the use of *capabilities*, which are objects for defining an indirect mapping between an activity and its required resource(s). Capability-based resource requirements permit the user to specify activity needs from an activity-based perspective and avoid having to assemble an individual set of resource types for each activity by hand.

Figure 2 illustrates the various relationships that can be defined in Comirem among activities, resources and capabilities, to facilitate a range of mappings between activities and their required resources. Using the more traditional approach, activities—both types and instances,

[2]The set of available resource instances in a situation is often referred to as the "bed-down" information.

[3]At the present time, Comirem does not support user-definable constraint types. The set of available constraint types is described in Section 3.2.3 (**Constraints**).

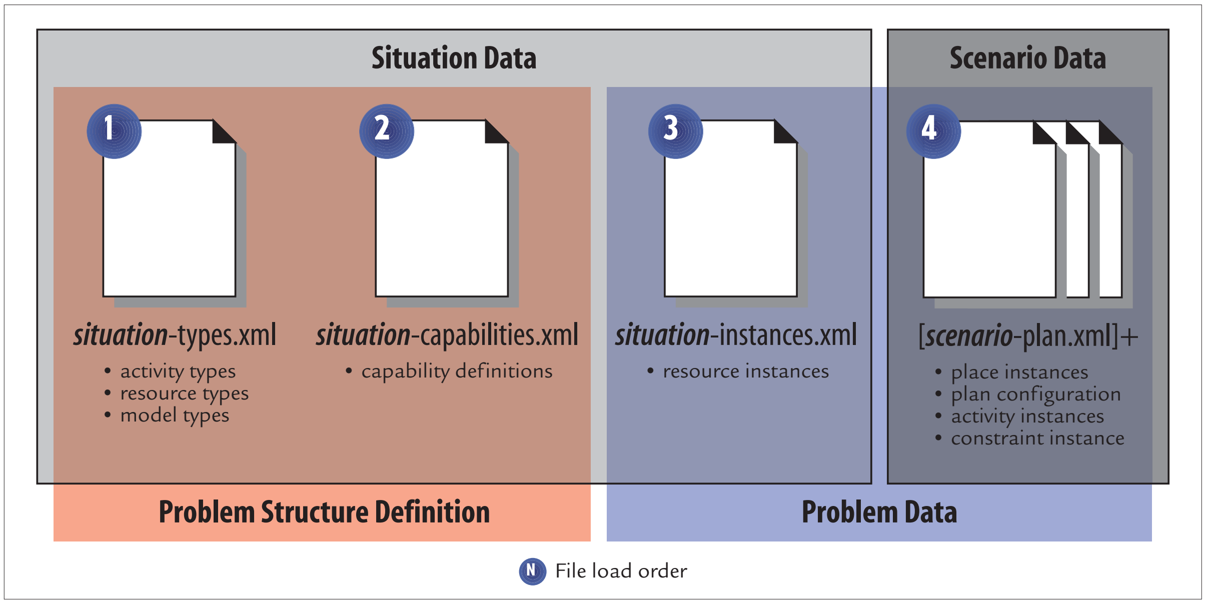


Figure 1: COMIREM input data sources

can be linked directly to a set of resources identified by either type, instance or model. Additionally, capabilities can be included in this mapping by allowing activities (again, both types and instances) to refer to required capabilities, which can in turn be linked to the resources that may satisfy their requirements (by type, instance or model). Finally, resources can also be linked to capabilities to signal their ability to deliver particular capabilities depending on how they are configured.[4]

In the remainder of this section, we introduce the primary components of the COMIREM conceptual model: activities (Section 3.2.1), resources (Section 3.2.2), constraints (Section 3.2.3), capabilities (Section 3.2.4) and plans (Section 3.2.5).

3.2.1 Activities

COMIREM provides two basic activity types, one for representing *moves* from one location to another (typically achieved using one or more mobile resources) and another for representing *events* taking place at a single location (and possibly requiring one or more stationary or mobile resources). Both of these activity types are illustrated in Figure 3. A third activity type, called a *wrapper*, provides a mechanism for enveloping other activities as in a parent-child relationship (with optional siblings) to represent hierarchical activity networks. A wrapper is an abstract activity whose semantics is more aptly defined by that of its constituent activities.[5]

Activities may impose a number of resource allocation constraints:

- *capability/resource requirements* designate the set of resources that might alternatively be used to support the activity, expressed either in terms of high-level *capabilities* (e.g., `light-`

[4]Resources refer to capabilities by means of a *configuration* mechanism that facilitates the representation of dynamically configurable resources that can be modified to deliver different capabilities over time. The configuration mechanism is described in Section 3.2.2.

[5]The concept of a *wrapper* activity is discussed in greater detail in Section 3.2.1.1.

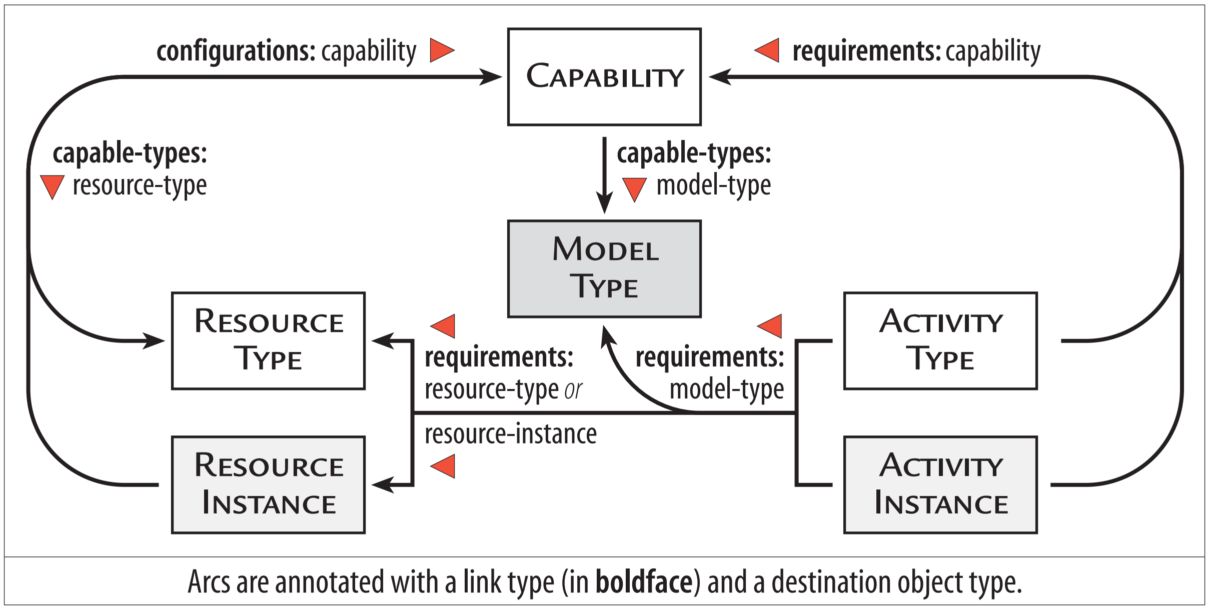

configurations: capability
Capability
requirements: capability
capable-types: resource-type
capable-types: model-type
Model Type
Resource Type
Activity Type
requirements: resource-type or resource-instance
requirements: model-type
Resource Instance
Activity Instance
Arcs are annotated with a link type (in boldface) and a destination object type.

Figure 2: Comirem conceptual model showing links between capabilities, resources (types & instances), resource model types, and activities (types & instances)

`transport`, `close-air-support`), specific *resource types* (e.g., `MH-47`, `C-130`, `HMMWV`, `MK-V`) or resource *model types* identifying relevant predefined resource classes (e.g., `ROTOR`, `FIXED-WING`, `WHEELED`, `TRACKED`), and can optionally dictate the exact number of resource instances required

- a *duration*, specified either as a range, a single fixed time value, or as a function of speed and distance (in the case of moves) that dictates how long the activity will take
- a *manifest*, specified in terms of common cargo and passenger types and quantities, indicating the capacity requirement of the activity

Absent an explicit designation of the number of resources required by an activity, the manifest is matched against the feasible configurations of a candidate resource to determine the

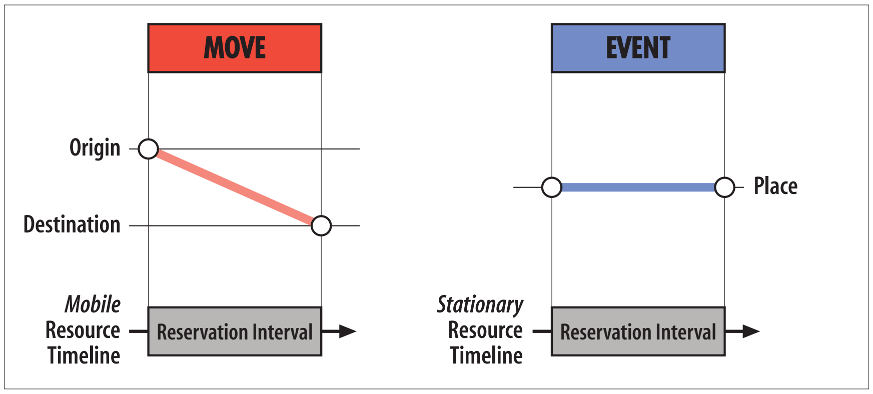


Figure 3: Move and event activity types

exact number of resource instances required to perform the activity.

3.2.1.1 Hierarchical Activity Networks

A given input activity may expand into a hierarchy of subactivities. For example, a move involving an aircraft may be comprised of positioning and depositioning legs, cargo-loading and unloading steps, and the actual flight leg between the origin and destination, all of which must be temporally synchronized. Activities with subactivities are referred to as *aggregate* activities and may be based on any of the COMIREM activity types: namely `move`, `event` or `wrapper`. COMIREM provides the means for specifying the expansion of an aggregate activity into a network of synchronized subactivities, with each subactivity possibly specifying its own requirements and imposing its own resource usage constraints, as illustrated in Figure 4.

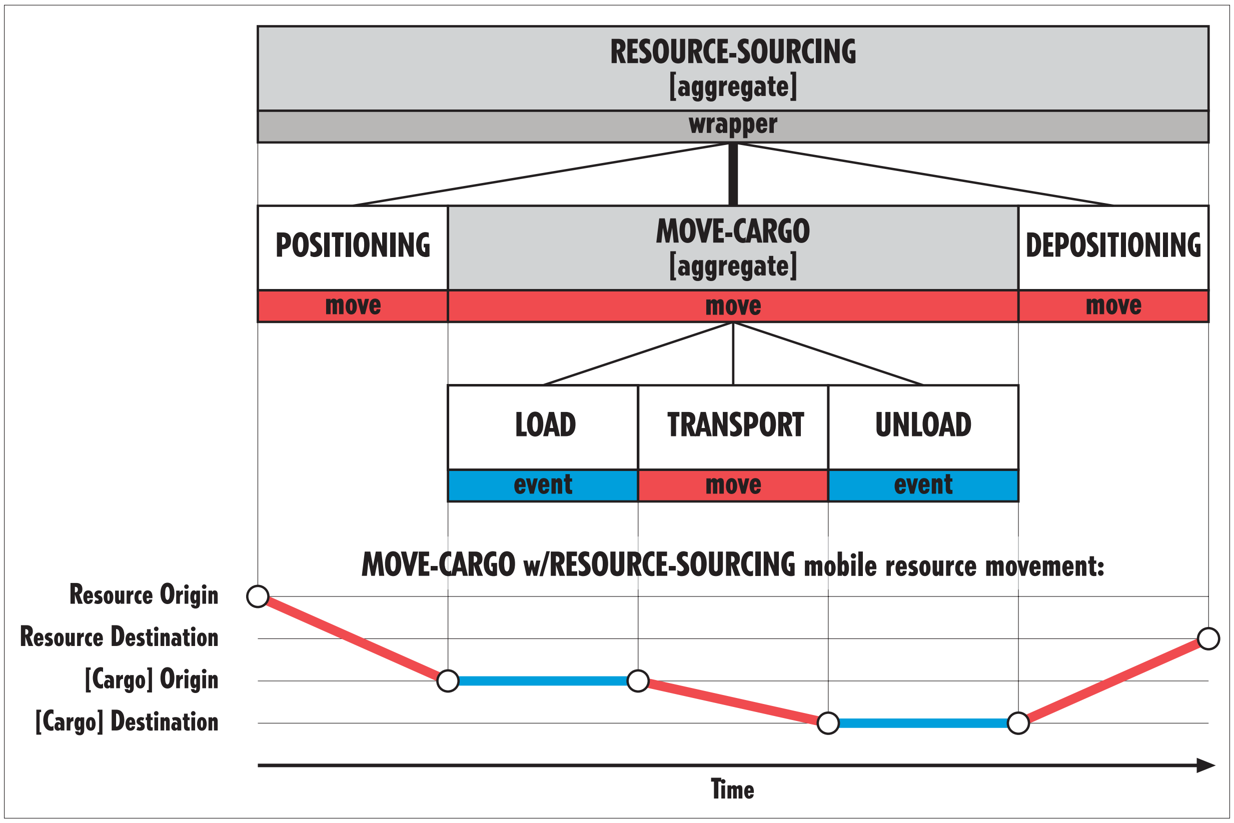


Figure 4: Aggregate activity type examples (`RESOURCE-SOURCING` and `MOVE-CARGO`)

The activity network hierarchy in Figure 4 defines three levels of abstraction for the movement of cargo using a mobile resource. At the top level, the resource must be *sourced*, or positioned, from a previous location and then delivered (or returned, as in the case of a round-trip reservation) to a subsequent location (note that one or both of these may be empty if the resource is already located at its designated origin or will be left at its designated destination). The `RESOURCE-SOURCING` aggregate activity identifies the overall extent during which a mobile resource must be reserved to perform its required tasks, including the positioning and depositioning and the actual cargo movement. The intermediate level shows the separation of the (de)positioning legs from the actual cargo-movement activity, MOVE-CARGO, which is itself an aggregate move activity. At the bottom level, the MOVE-CARGO activity expands into three basic steps: a cargo-loading event that allocates time for cargo to be loaded onto the mobile resource, the actual move activity during which the cargo will be transported, and the final cargo-unloading event. At the bottom of Figure 4, note that the location of a mobile resource assigned to the RESOURCE-SOURCING activity is tracked over time, in conjunction with the leaf nodes in the activity network.

The specification of activity networks in COMIREM relies on the concept of an anchor, or *nexus* activity, which identifies the level of detail at which a network of activities is defined within a plan. The nexus activity is the activity within a hierarchical activity network (as shown in Figure 4) from which the entire (local) network is derived, in both an upward and downward direction. It provides a convenient handle to the activity network that can be easily referenced by other activities within a plan, and thereby helps to collectively define the level at which an entire plan is most appropriately viewed.

For example, the activity network in Figure 4 is arguably most logically anchored by the MOVE-CARGO activity, because it identifies the logical scope for a cargo-movement activity in a typical logistical plan. From the perspective of the planner, the movement of cargo from one location to another represents a primary action in such a plan. The sourcing of the mobile resources used to move the cargo is a supporting action that can effectively be handled in the background. Synchronization among activities in a plan will most likely be specified in terms of cargo movements, ignoring any potential supporting (de)positioning legs that may need to be instantiated to facilitate them.

Given this view, the (de)positioning legs surrounding the MOVE-CARGO activity can be viewed as providing a *wrapper* around the activity that defines the necessary positioning activities of any mobile resource secured for the task. Figure 5 illustrates the recursive expansion of a nexus activity both downward into child subactivities and upward into wrapper activities. Note that any activity within an expanding activity network, be it derived as a child or wrapper activity, may itself further expand into child or wrapper activities. Returning again to Figure 4 and using MOVE-CARGO as the nexus activity, it should be evident that the RESOURCE-SOURCING aggregate activity is instantiated as a wrapper around MOVE-CARGO, with the POSITIONING and DEPOSITIONING moves as its children, and that the LOAD, TRANSPORT and UNLOAD events are instantiated as the direct children of the MOVE-CARGO aggregate activity.

One additional definition is appropriate at this point. We use the term *derivative* to describe an optional activity (i.e., an activity whose duration may be zero). Positioning legs for mobile resources are the most obvious derivative activities. If a mobile resource is required to be at a specific location to load cargo for transport and is already collocated with the cargo, then the duration of a positioning leg to get it from its current location to the location of the cargo is zero, and the activity is therefore a no-op. It only warrants a non-zero duration if the resource must move between distinct locations, in which case the duration is dependent on its speed and the necessary travel distance.[6]

COMIREM relies on *variables* within activity networks to both retain and convey values across individual activities. Variable values are stored in class slots in activity instances and passed to child and wrapper activities during activity network instantiation. Two typical variables used to

[6]All distances in COMIREM are derived using a Great Circle calculation.

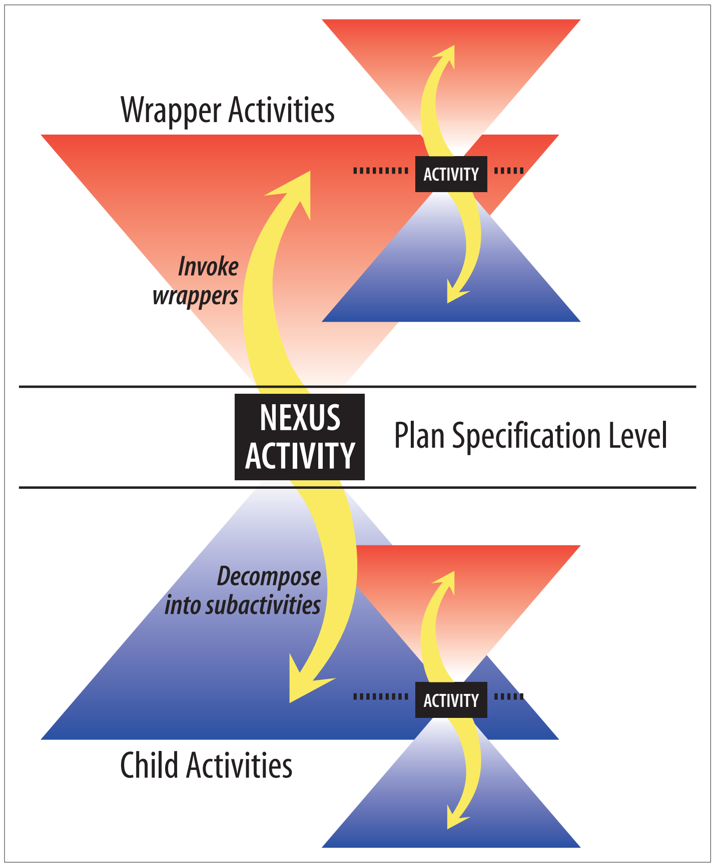


Figure 5: Instantiation of wrapper and child activities from a nexus activity

support mobile resource sourcing are `resource-origin` and `resource-destination`. These variables allow the planner to indicate a specific location from which to secure a resource and a specific location to which it should be returned following its use, by passing their values to relevant supporting move activities. To define round-trip sourcing, the `resource-origin` and `resource-destination` variables are simply set to the same location. In a more flexible plan, where the sourcing locations of resources may not be specified, these variables can be initialized as *free* variables, thereby allowing the scheduler to control their assignment in the process of considering alternative resource options. Resources can therefore be obtained from, and returned to, any location(s) that can be reached in the amount of time that is available both prior to, and following, the activity in question.[7]

To further illustrate the concept of activities in Comirem, we provide (in the following three figures) annotated snapshots of XML data files that demonstrate how some of the entities we have addressed in this section can be specified. Figure 6 presents the definition for the `RESOURCE-SOURCING` aggregate wrapper activity. Figure 7 presents the definition for the `MOVE-CARGO` aggregate activity that references the `RESOURCE-SOURCING` wrapper. Both of these definitions belong in the *situation*`-types.xml` data file. Finally, Figure 8 demonstrates how an instance of a predefined activity type is specified, in this case by referencing the `MOVE-CARGO` activity. This

[7]The process of binding free variables is described in detail in Section 3.2.1.4.

definition belongs in the *scenario*-`plan.xml` data file.[8]

In Figure 6, the RESOURCE-SOURCING wrapper adds positioning and depositioning move activities around an invoking child activity (e.g., MOVE-CARGO from Figure 4) in a directly abutting sequence and passes along the necessary variable values (i.e., `resource-origin`, `resource-destination`, `origin` and `destination`) to help in determining required travel distances. The distinction between `<variables>` and `<inputs>` is that *variables* lacking values at instantiation time are considered free, while *inputs* must be provided by the invoking activity. Because this activity is intended to be applicable to a wide range of resource types, requirements for the specific mobile resource(s) to be sourced are specified with the invoking child activity instance, in which case, as a result of this wrapper, the necessary time block for the reservation(s) will include the (de)positioning legs.

```
<type>
  <wrapper name="resource-sourcing">
```

Declare the variables to be maintained by the activity:

```
    <variables>resource-origin resource-destination</variables>
```

Declare the variables required from the invoking child activity:

```
    <inputs>origin destination</inputs>
```

Define the decomposition of an activity into subactivities and specify how values are passed down from parent to child. The `initarg` forms specify how values are assigned to subactivity variables. For example, the `origin` and `destination` of the ***positioning*** move are assigned the **resource-sourcing** activity's `resource-origin` and `origin` slot values, respectively):

```
    <decomposition>
      <child-spec name="positioning">
        <type-ref>move</type-ref>
        <initargs>
          <initarg name="origin">
            <slot>resource-origin</slot>
          </initarg>
          <initarg name="destination">
            <keyword>origin</keyword>
          </initarg>
        </initargs>
      </child-spec>
      <child-spec name="depositioning">
        <type-ref>move</type-ref>
        <initargs>
          <initarg name="origin">
            <keyword>destination</keyword>
          </initarg>
          <initarg name="destination">
            <slot>resource-destination</slot>
          </initarg>
        </initargs>
      </child-spec>
    </decomposition>
```

Define the temporal sequencing of subactivities (`:child` refers to the subactivity that invoked the wrapper). The default lower bound for `successor` constraints is 0 minutes, so the additional zero-duration upper bounds indicate that these activities must abut one another:

```
    <constraints>
      <constraint>
        <type-ref>successor</type-ref>
        <from>positioning</from>
        <to>:child</to>
        <ub><interval><minutes>0</minutes></interval></ub>
      </constraint>
      <constraint>
        <type-ref>successor</type-ref>
        <from>:child</from>
        <to>depositioning</to>
        <ub><interval><minutes>0</minutes></interval></ub>
      </constraint>
    </constraints>

  </wrapper>
</type>
```

Figure 6: Annotated XML specification for the RESOURCE-SOURCING wrapper activity type

[8]While the RESOURCE-SOURCING aggregate activity is not currently part of the COMIREM activity type library, it is a logical candidate for a SOF-specific activity-type library, as we discuss in Section 6.1.1 (***Plug-In* Domain Models**).

In Figure 7, the `MOVE-CARGO` aggregate activity is defined as a `move` activity that expands both downward and upward to generate the activity network shown in Figure 4. It decomposes into three subactivities (of predefined types not shown) that divide the overall cargo-movement action into separate loading, transporting and unloading activities. It invokes the `RESOURCE-SOURCING` wrapper to provide the necessary (de)positioning legs, and passes on the necessary `resource-origin`, `resource-destination`, `origin` and `destination` variable values to help determine the required travel distances for those moves. The former two variables are defined as inputs to be specified with an instance of this type (or left unspecified to signify free variables), while the latter two reference the standard `move` slots. As in the previous example, there are no resource requirements specified with this type definition, so that it may be used to define a variety of cargo-movement activities. The requirements instead should come from the instance definition that references this activity type (see Figure 8).

Finally, in Figure 8, an instance of the `MOVE-CARGO` activity type is defined to move 120 passengers (PAX) from `home-station` to `staging-area`. Two aspects of this definition are worth highlighting:

1. Values for the `resource-origin` and `resource-destination` variables maintained by the `MOVE-CARGO` move activity and `RESOURCE-SOURCING` wrapper activity are specified explicitly using `initarg` tags, thereby requiring that any resource allocated to perform this task must be located initially at `home-station` (thus eliminating the need for a positioning leg) and then returned to `home-station` as part of the unsourcing actions (i.e., the depositioning leg). If not specified, the scheduler is free to secure a mobile resource from any location—as long as it can be sourced to `home-station` within the available time window—and to leave the resource at the destination of the cargo move, namely `staging-area`.

2. The resource requirements for this activity instance are specified in terms of a capability, namely `light-transport`, indicating that the scheduler is free to consider any available resource belonging to a class defined as capable of performing a `light-transport` kind of task.[9]

[9] Comirem treats `requirement` entries as *parallel* resource requirements, meaning that additional resources must be secured for the same time period—i.e., the scope of the requirement (in this case, the `position-for-insertion` activity). Note that additional actions specified by wrapper activities (such as sourcing moves) may result—legitimately—in different overall reservation durations.

```
<type>
  <move name="move-cargo">
```

Declare the variables to be maintained by the activity:

```
    <variables>resource-origin resource-destination</variables>
```

Define the decomposition of an activity into subactivities and specify how values are passed down from parent to child. The `initarg` forms without `slot` attributes (i.e., in the ***transport*** subactivity) indicate that values are assigned to the subactivity's variables using the identically named variables of the parent:

```
    <decomposition>
      <child-spec>
        <type-ref>load</type-ref>
        <initargs>
          <initarg name="place">
            <slot>origin</slot>
          </initarg>
        </initargs>
      </child-spec>
      <child-spec>
        <type-ref>transport</type-ref>
        <initargs>
          <initarg><slot>origin</slot></initarg>
          <initarg><slot>destination</slot></initarg>
        </initargs>
      </child-spec>
      <child-spec>
        <type-ref>unload</type-ref>
        <initargs>
          <initarg name="place">
            <slot>destination</slot>
          </initarg>
        </initargs>
      </child-spec>
    </decomposition>
```

Define the temporal sequencing of subactivities:

```
    <constraints>
      <constraint>
        <type-ref>successor</type-ref>
        <from-type>load</from-type>
        <to-type>transport</to-type>
        <ub><interval><minutes>0</minutes></interval></ub>
      </constraint>
      <constraint>
        <type-ref>successor</type-ref>
        <from-type>transport</from-type>
        <to-type>unload</to-type>
        <ub><interval><minutes>0</minutes></interval></ub>
      </constraint>
    </constraints>
```

Define the wrapper activities invoked by an activity and specify how values are passed up from child to (wrapper) parent:

```
    <wrappers>
      <wrapper name="resource-sourcing">
        <initargs>
          <initarg><slot>origin</slot></initarg>
          <initarg><slot>destination</slot></initarg>
          <initarg>
            <slot>resource-origin</slot>
          </initarg>
          <initarg>
            <slot>resource-destination</slot>
          </initarg>
        </initargs>
      </wrapper>
    </wrappers>
  </move>
</type>
```

Figure 7: Annotated XML specification for the MOVE-CARGO activity type

```
<instance>
  <activity name="position-for-insertion">
```

Indicate the type of activity to instantiate:

```
    <type-ref>move-cargo</type-ref>
```

Provide initial slot values for the activity:

```
    <description>position insertion teams at staging area</description>
    <origin>home-station</origin>
    <destination>staging-area</destination>
```

Provide initial slot values for the attributes of any derived activities (i.e., from either wrappers or subactivities):

```
    <initargs>
      <initarg name="resource-origin">
        <value>home-station</value>
      </initarg>
      <initarg name="resource-destination">
        <value>home-station</value>
      </initarg>
    </initargs>
```

Specify the resource requirements for the activity (in this case, via a capability):

```
    <requirements>
      <requirement>
        <capabilities>light-transport</capabilities>
      </requirement>
    </requirements>
```

Specify the manifest for the activity:

```
    <manifest>
      <manifest-entry>
        <cargo count="120">PAX</cargo>
     </manifest-entry>
    </manifest>

  </activity>
</instance>
```

Figure 8: Annotated XML specification of a MOVE-CARGO activity instance

3.2.1.2 Activity Attributes

Activity attributes can be specified at either the class or instance level, depending on the preferences of the domain modeler and the nature of the domain. Class attributes define a default value for all class instances, while instance attributes permit an instance to override a class-level default with its own value. A typical example of this issue involves the speed of a resource. A particular type of resource (e.g., MH-60) may have a standard average traveling speed of 177 mph, while a particular MH-60 instance (e.g., MH-60-001) may have been altered to increase its average traveling speed to 200 mph. By specifying a speed attribute with a value of 177 mph in the MH-60 resource-class definition form, all MH-60 instances will—by default—inherit the 177 mph average traveling speed value. In the case of MH-60-001, specifying a speed attribute with a value of 200 mph in its instance definition form will allow it to override the default 177 mph value with its own individual 200 mph value.

In the activity attribute tables that follow (Tables 1 and 2), the various activity attributes are classified according to how they are *generally* specified. But note that in practice they could be specified at either the class or instance level. Note also that the attributes in these tables follow the XML grammars described in Section A.2.

Table 1: Activity type attributes

- *Name* [**required**]

  The name of the activity type

- *Variables* [optional]

  The variables maintained by an activity, typically for use in informing the duration-calculation process (e.g., for dealing with mobile-resource positioning)

  For wrappers, variable values are provided by the invoking activity; otherwise, they are provided by the activity instance. If left unspecified, they are considered free.

- *Inputs* [optional]

  The variables to be initialized at instantiation time by an activity instance

- *Description* [optional]

  A description of the activity

- *Decomposition* [optional]

  A template that defines how an activity is expanded into constituent subactivities

- *Constraints* [optional]

  The temporal sequencing constraints that govern any constituent subactivities

  Within this specification, a special `:child` variable can be used to reference the invoking activity during instantiation.

- *Wrappers* [optional]

  The wrapper-activity types to be invoked during instantiation

- *Requirements* [optional]

  Specifies the resources or capabilities required by an activity

- *Derivative-p* [optional]

  A flag indicating whether or not the activity type is derivative in nature (i.e., its instances only become necessary as the result of some other scheduling decision, as with a (de)positioning leg for a mobile resource) *The default is nil*.

- *Capability-Type* [optional]

  Indicates what kind of capabilities are appropriate for this activity (i.e., `move` or `event`) *The default is obtained from the activity's base type*.

  This value is generally only overridden in special cases (e.g., when a mobile resource is required by an `event`, in which case `move`-specific capabilities may be more appropriate)

Table 2: Activity instance attributes

- *Name* [**required**]

  The name of the activity

- *Type-Ref* [**required**]

  The type of the activity: either `move`, `event` or `wrapper`, or a user-defined activity type

- *Display-Name* [optional]

  A presentable print-name for the activity (for use in the GUI)

- *Origin* [**required**]

  For moves, indicates the origin location

- *Destination* [**required**]

  For moves, indicates the destination location

- *Place* [**required**]

  For events, indicates the location at which the event takes place

- *Initargs* [optional]

  A collection of keyword-value pair specifications for passing values either down to subactivities, or up to wrappers, at the point of instantiation

- *Manifest* [optional]

  Describes the cargo associated with an activity, in terms of predefined cargo types (e.g., PAX (personnel), `MUNITIONS`, `PALLETS`, `FUEL`, `AIRCRAFT`)

- *Duration* [optional]

  The duration of an activity, specified as either a minimum, maximum, min/max pair, or fixed value

- *Time-Bounds-Constraints* [optional]

  Specifies (1) any earliest/latest start/finish-time anchoring constraints (e.g., EST, LST, EFT, LFT) on an activity using fixed temporal values, and (2) the ref-hours to which an activity may be linked, by either its start or finish time, or both

3.2.1.3 Sibling Activities

It is often the case that an activity requires more than a single resource instance—or collection of single resource instances—to complete its desired task, generally depending on the size and nature of its manifest.[10] For example, consider that an activity responsible for moving personnel from one location to another and requiring instances of the `MH-60` helicopter class needs enough `MH-60` instances to accommodate the number of personnel involved (according to the activity's manifest). In these cases, COMIREM uses *sibling* activities to ensure that the proper number of resource instances are allocated to an activity. A sibling is a clone of an original nexus (i.e., plan-level) activity—and its fully expanded network of wrapper and child activities, which helps transform a single activity, in a *copy-and-paste* fashion, into a series of multiple identical and possibly parallel activities, each requiring a single resource instance (or set of resource instances).

Owing to their support of flexible decision-making, siblings are ephemeral in nature, lasting only for as long as the planner/user stays with a particular resource requirement and/or manifest configuration, or actual resource assignment.[11] If either of an activity's requirements or manifest are changed at any time, then the number of siblings required by that activity may also change in response.

In Figure 9, a fragment of a typical plan is presented. Prior to scheduling, Activity `A3` requires some number of instances of the `MH-60` helicopter class to handle 48 `PAX` (personnel). At the

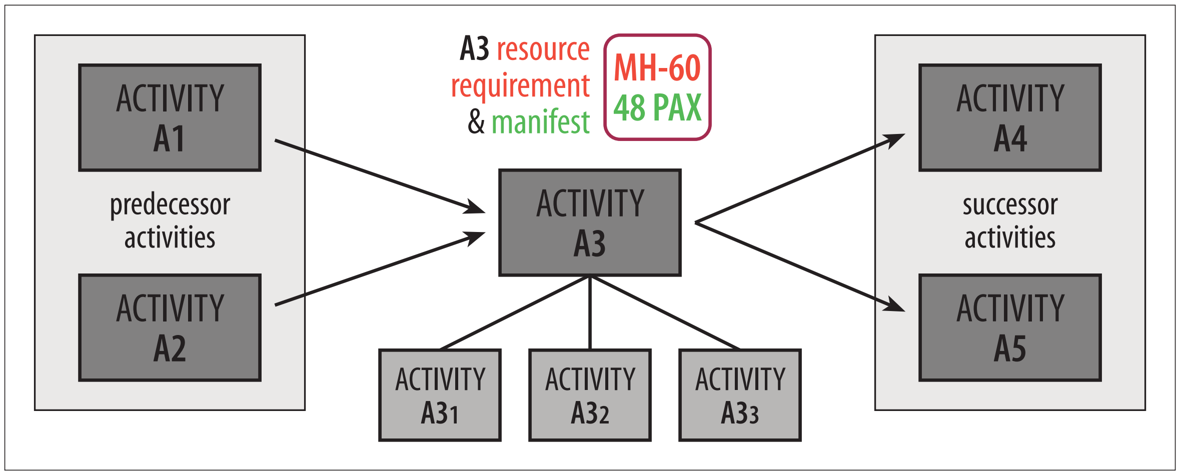


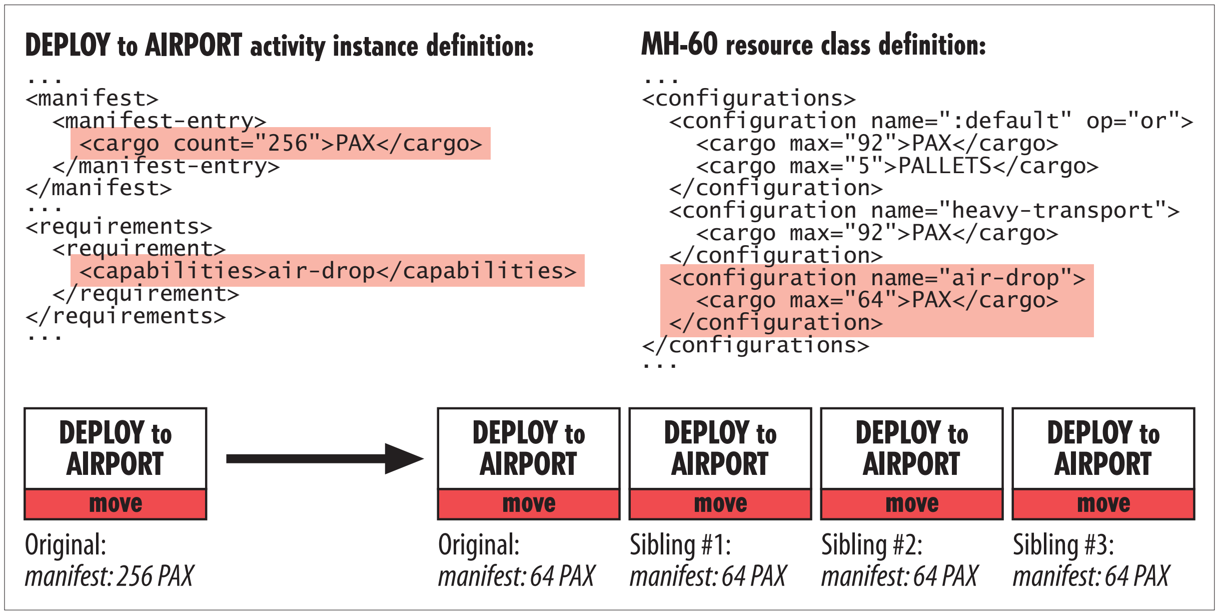
Figure 9: Fragment of a plan network (prior to sibling creation)

point when the scheduler begins allocating resources to Activity `A3`, it first determines, based on its knowledge of the capacity of the relevant `MH-60` configuration required for this task (i.e., 24 `PAX`), that two `MH-60`s are needed. This determination process is illustrated in Figure 10. A single sibling activity network is then instantiated and spliced into the network, as shown in Figure 11 (the sibling and new temporal sequencing constraints are highlighted in red). If an

[10]Recall that the planner may also declare exactly how many instances of a resource are required by an activity, thereby overriding the manifest-based calculation.

[11]In keeping with this fact, the COMIREM GUI will display siblings—only if they are scheduled at times different from their original activities—but they cannot be directly manipulated by the user. If a sibling is selected in the GUI, COMIREM will treat it as a selection of the original activity.

assignment of MH-60s is made at this point, the overall plan network will stay in this state until the allocation is changed. If an assignment is not made, the sibling will be removed and deleted, and the network will return to its original state.



DEPLOY to AIRPORT activity instance definition:

```
...
<manifest>
  <manifest-entry>
    <cargo count="256">PAX</cargo>
  </manifest-entry>
</manifest>
...
<requirements>
  <requirement>
    <capabilities>air-drop</capabilities>
  </requirement>
</requirements>
...
```

MH-60 resource class definition:

```
...
<configurations>
  <configuration name=":default" op="or">
    <cargo max="92">PAX</cargo>
    <cargo max="5">PALLETS</cargo>
  </configuration>
  <configuration name="heavy-transport">
    <cargo max="92">PAX</cargo>
  </configuration>
  <configuration name="air-drop">
    <cargo max="64">PAX</cargo>
  </configuration>
</configurations>
...
```

Figure 10: Sibling requirement calculation using manifest quantity and resource configuration information

Note that while the splicing of siblings is done in a manner that facilitates parallel processing of the original activity and its siblings, parallelism is not guaranteed. If only a single resource is available (e.g., MH-60-001 in the case above), then the original activity and its sibling will have to be performed in sequence. Figure 12 illustrates the case involving an activity and three siblings scheduled on either one, two, three or four resources. A strict sequential ordering results from the one-resource case (**[a]**), while complete parallelism is achieved when four resource instances are available (**[d]**).

Finally, there are some conceptual limitations on siblings that are worth noting:

- The current model assumes that the changing of a resource's configuration is a zero-duration setup operation.[12] This is because the scheduler looks to the resource class to assess the capacity of a resource in a particular configuration instead of checking the configuration of each individual candidate resource at the time it is expected to be used.

- Siblings inherit the identical resource requirements from their original activity and must *all* be scheduled using the same specification (though *not* the same instance(s)). For example, if the user selects the AC-130H aircraft type for an activity from among a list of possible aircraft types, and the scheduler determines that three AC-130H aircraft instances are required to perform the task, then each of the original activity and its two siblings must be assigned a single AC-130H aircraft instance (i.e., there can be no *mix-and-match* making

[12]To be more precise, COMIREM resources currently do not change their configuration over time, although this capability *is* envisioned (and partially supported).

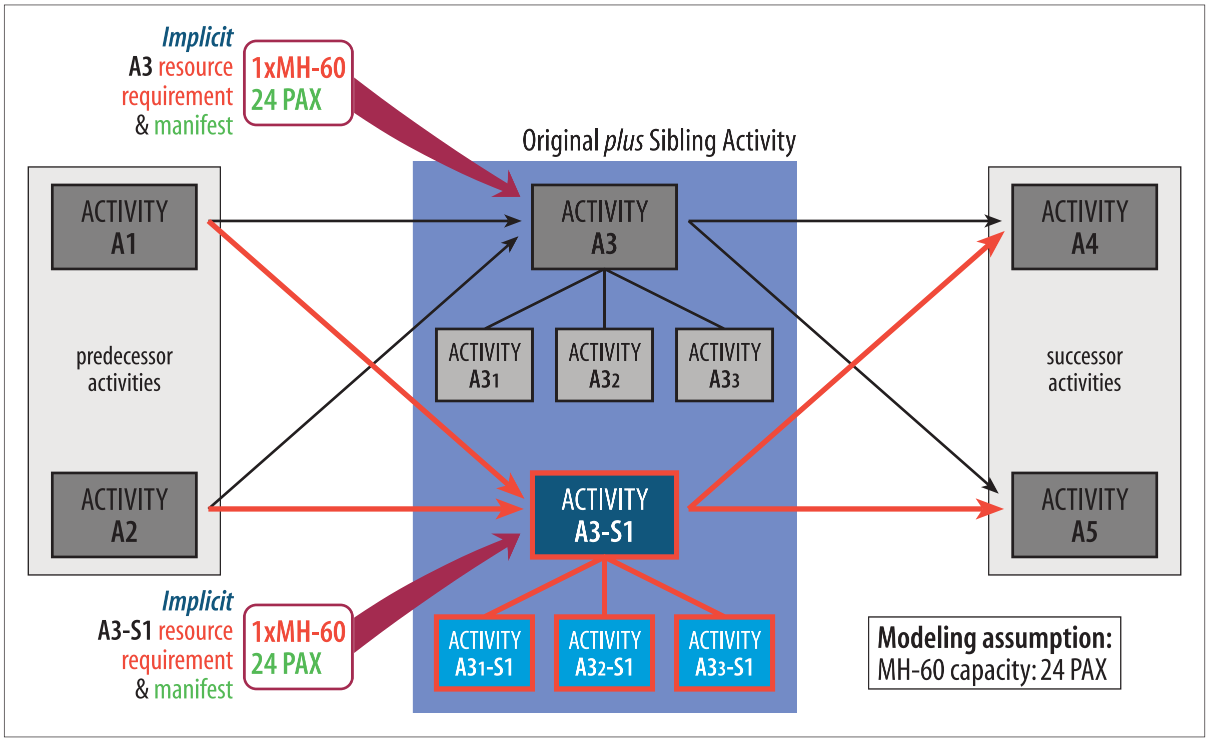


Figure 11: Fragment of a plan network following the instantiation of sibling activities

use of other possible required resources). This is because siblings are instantiated and scheduled by the scheduler in the same step as their original activity, and only a single calculation is performed to determine the required number of siblings based on a particular resource type that applies to all activities (original and siblings) at the same time.

By extension, this limitation also applies to the case where multiple resources are required for an activity by virtue of wrapper or child activities that may add their own individual resource requirements to those of the nexus activity. In these cases, the scheduler creates a single collective resource requirement at the outset of the scheduling step, determines the number of siblings required, and uses the same resource set for each activity network anchored by the original and its siblings.

- Related to the previous item, activities and their siblings also require that sourced mobile resources (i.e., those requiring (de)positioning legs) report from and return to the same location as specified by the original activity instance. This means that there is no optimization done to minimize the use of extraneous positioning legs.

  Note that this is *not* the case when specific sourcing locations, such as the `resource-origin` and `resource-destination` variables mentioned earlier, are unspecified (i.e., free).

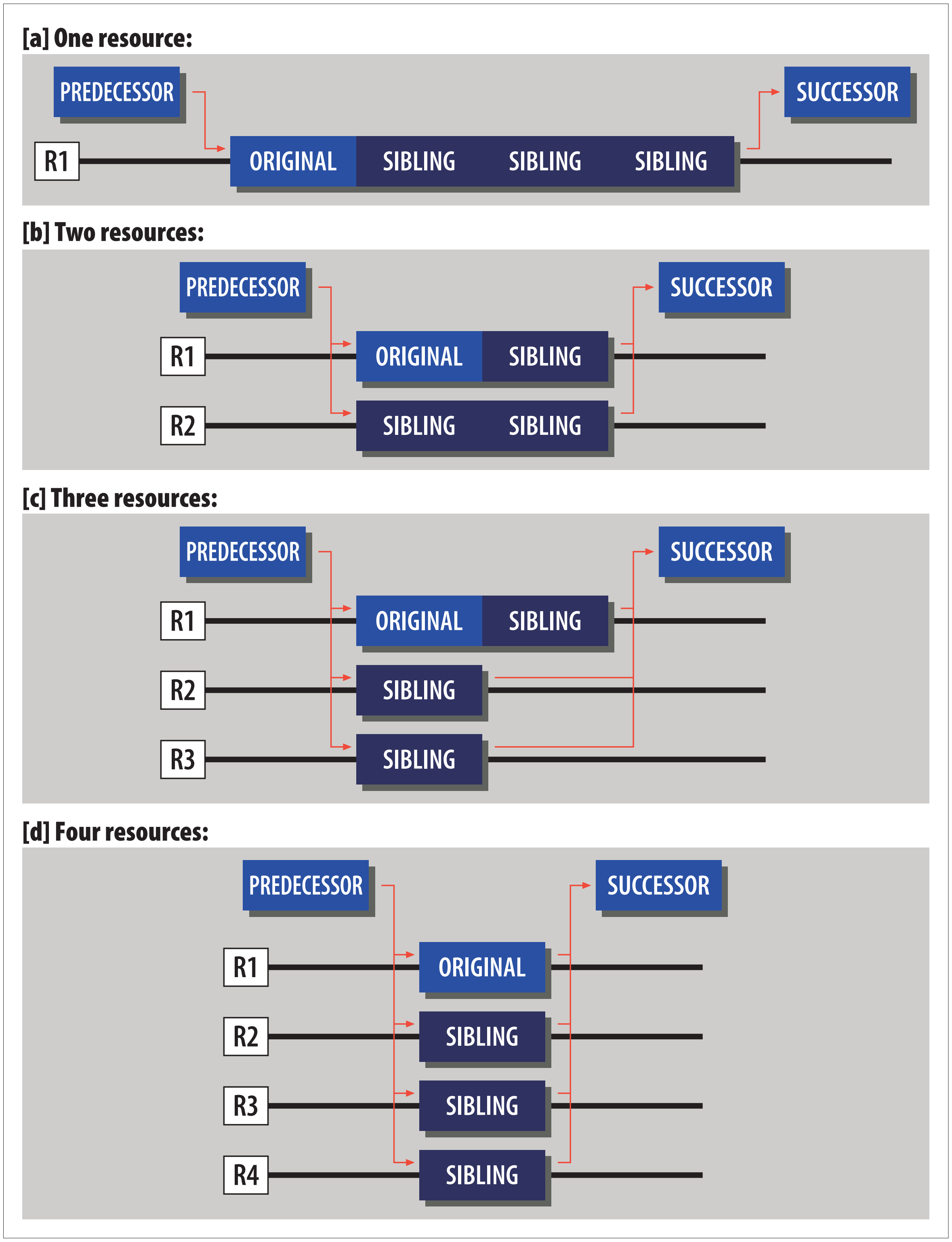


Figure 12: Sequencing possibilities for sibling activities: example scenario consisting of three siblings and an original activity scheduled on one, two, three and four resources

3.2.1.4 Resource Installation

To maintain flexibility in the evolving plan network throughout the resource-allocation process, Comirem implements a least-commitment, flexible time-bounds scheduling approach that delays the refinement of task durations for as long as possible and avoids completely a strict temporal sequencing of resource assignments (this latter process is described in Section 3.3). In this section, we focus on the former process, which we refer to as *resource installation*, and describe its purpose, its behavior, and its implications.

Because Comirem supports the specification of multiple resource-requirement options for a single activity, and because the duration of an activity may depend on the particular characteristics of the resource allocated to it (owing to either the speed of the resource or some other relevant performance attributes), Comirem needs a mechanism for tightening its underlying plan representation when a scheduling decision is made that narrows the field of feasible candidates—and therefore more accurately assesses the temporal requirements—for an activity.

As an example, consider a logistical environment that involves the movement of cargo from location to location, with the possibility of using multiple types of mobile resources for the movement activities. The duration of each movement is a factor of (1) the distance to be traveled from origin to destination and (2) the speed of the mobile resource assigned to perform the task. At first, the duration of each such movement activity A is a range of the form: $[\boldsymbol{lb}(A), \boldsymbol{ub}(A)]$, where $\boldsymbol{lb}(A)$ is the speed of the fastest candidate resource type and $\boldsymbol{ub}(A)$ is the speed of the slowest candidate resource type for activity A. It is not until the point at which the scheduler allocates a specific resource instance to a movement task that its duration converges on a fixed value $\boldsymbol{D}(A) : \boldsymbol{lb}(A) \leq \boldsymbol{D}(A) \leq \boldsymbol{ub}(A)$, determined by the speed of the selected resource and the travel distance required. Once $\boldsymbol{D}(A)$ has been determined, the duration of the movement task can be fixed and the network can be tightened to reflect the assignment. (But note that this fixed-duration activity can still float within its possibly flexible time bounds, depending on the available slack in its own plan.)

Whenever a specific resource candidate is considered for an activity, either as part of the process of determining feasible scheduling options or performing an allocation, Comirem must first install the resource on the activity to understand the exact temporal requirements of the activity when paired with the resource in question. The basic installation process is summarized in Table 3.

Figure 13 presents a graphical summary of an example situation existing prior to resource installation when there is a choice between two resource types to perform a typical activity, and Figure 14 illustrates the possible outcomes of the resource-installation process depending on which resource type is chosen. Referring to Figure 13, notice that there are `MH-60` helicopters located at `Resource Location 1`, and `MH-47` helicopters located at `Resource Location 2`. The activity to which one of these resource types must be assigned is a roundtrip move from the `Staging Area` to the `Enemy Base`, with book-ended positioning and depositioning legs to source the aircraft. If the `MH-60` resource is chosen, the resources will be sourced from and returned to `Resource Location 1`. If the `MH-47` resource is chosen, the resources will be sourced from and returned to `Resource Location 2`. The actual required duration of the movement activity is either 175 minutes using the `MH-60` resource to travel 537.5 miles at 184 mph, or 212 minutes using the `MH-47` resource to travel 625 miles at 177 mph.

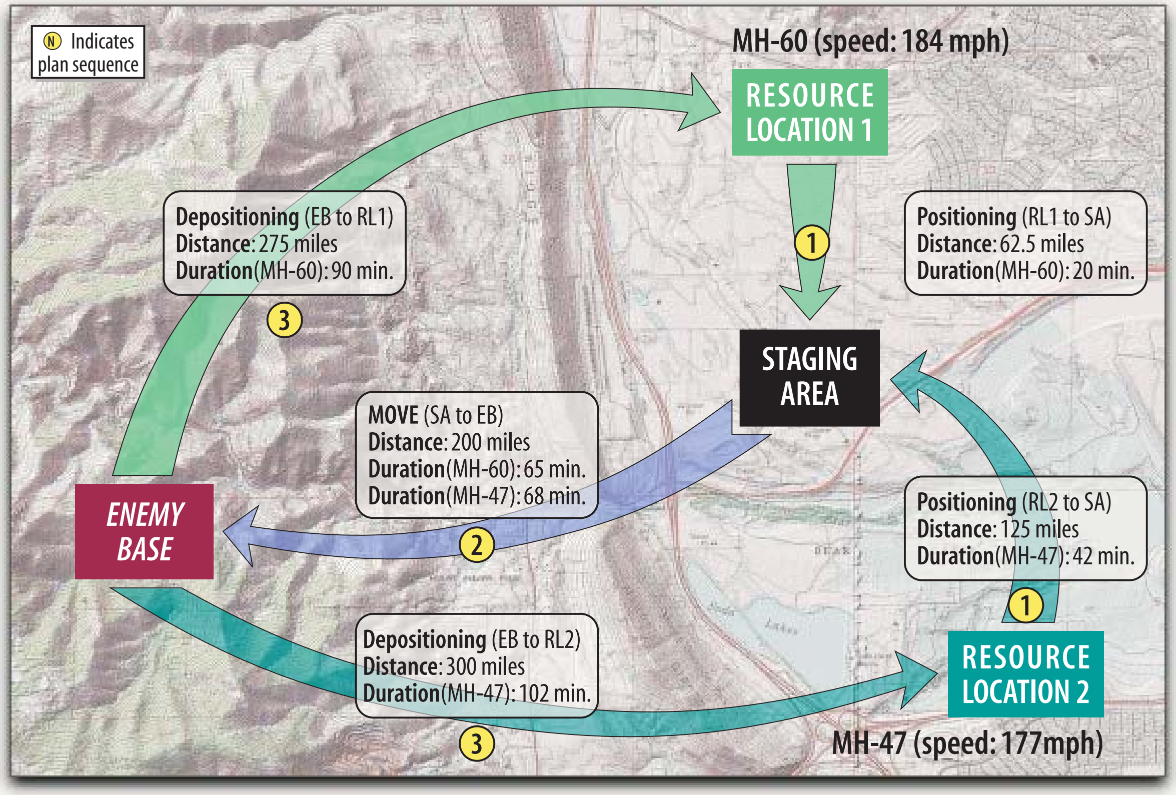
Referring now to Figure 14, notice the state of the movement activity prior to resource instal-

Table 3: Resource installation process description

```
Operation-Install-Resources:

    With resource as input:

    [1] Bind free variables
    [2] Call Operation-Install-Resources recursively on all subactivities:

        For each leaf-level subactivity,
        [a] Cache existing temporal bounds
        [b] Calculate and assert new temporal bounds
            given resource and all relevant free variable bindings
        [c] Check network consistency
            [i]  In case of inconsistency, undo all changes and throw a failure
            [ii] Otherwise, return success

    [3] If a failure is caught, undo all changes and throw a failure
        Otherwise, return success
```

lation. Based on the two resource options, the move will take either 65 or 68 minutes, and the (de)positioning legs are empty. Following the installation of the `MH-60` resource, whereupon the `Resource Origin` and `Resource Destination` variables are bound to `Resource Location 1`, the exact temporal requirement for the activity and its (de)positioning legs is shown in **Case 1**. Following the installation of the `MH-47` resource, with the `Resource Origin` and `Resource Destination` variables bound to `Resource Location 2`, the exact temporal requirement for the activity and its (de)positioning legs is shown in **Case 2**.

As long as a resource is installed on an activity, whether or not it is allocated to that activity, the temporal constraints it imposes on the network remain in effect. Only after uninstalling the resource—a process that reverses the actions of the installation by restoring the cached original temporal bounds for all affected activities—is the original temporal flexibility of the activity restored to the network.



Resource sourcing options for MOVE(Staging Area to Enemy Base):
- MH-60 from Resource Location 1
- MH-47 from Resource Location 2

Figure 13: Resource installation example: a scenario

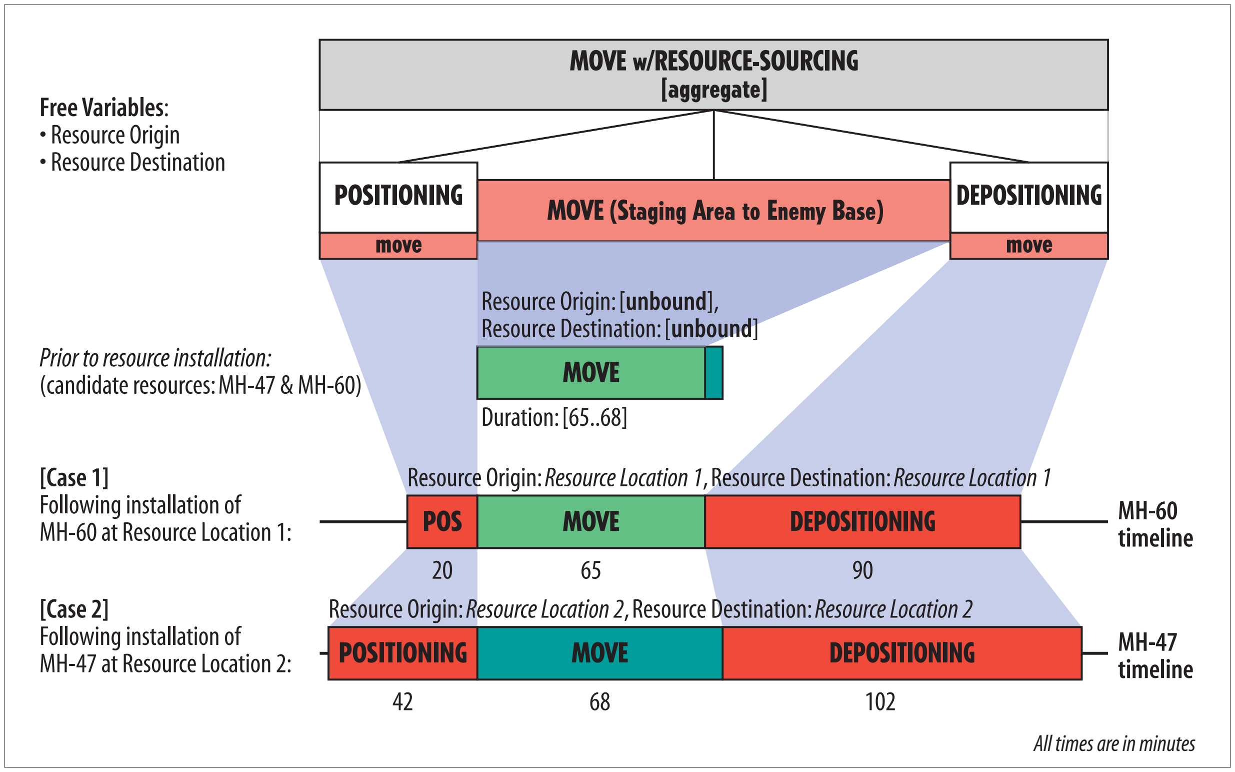

MOVE w/RESOURCE-SOURCING [aggregate]
Free Variables:
• Resource Origin
• Resource Destination
POSITIONING
move
MOVE (Staging Area to Enemy Base)
DEPOSITIONING
move
Resource Origin: [unbound],
Resource Destination: [unbound]
Prior to resource installation:
(candidate resources: MH-47 & MH-60)
MOVE
Duration: [65..68]
[Case 1]
Following installation of
MH-60 at Resource Location 1:
Resource Origin: Resource Location 1, Resource Destination: Resource Location 1
POS
MOVE
DEPOSITIONING
MH-60 timeline
20
65
90
[Case 2]
Following installation of
MH-47 at Resource Location 2:
Resource Origin: Resource Location 2, Resource Destination: Resource Location 2
POSITIONING
MOVE
DEPOSITIONING
MH-47 timeline
42
68
102
All times are in minutes

Figure 14: Resource installation example: temporal implications of two resource alternatives

3.2.1.5 Threads

The remaining activity network structure provided by Comirem is called a *thread*, which facilitates the aggregation of a sequence of temporally and geographically contiguous activities so that they may share a set of resources amongst themselves without having to allocate them each individually. It is often the case that a series of activities may require the same resource or collection of resources to be used by each constituent activity, as when a military taskforce is configured and assigned to perform an entire mission or some portion of a mission, and is also responsible for providing its own equipment. Threads effectively define a local context within which a set of resources can be shared by a group of activities. During the time when resources are assigned to a thread, they are available exclusively to the activities that comprise the thread.[13] The basic thread structure is presented in Figure 15.

The constraints on thread creation are as follows:

- Activities must abut one another; the temporal sequencing constraint (i.e., *successor*, *before*) between each activity must have temporal bounds of [0,0]
- The sequence of activities must reflect a continuous route from one specific location to another, as specified by a move's *origin* and *destination* or an event's *place*

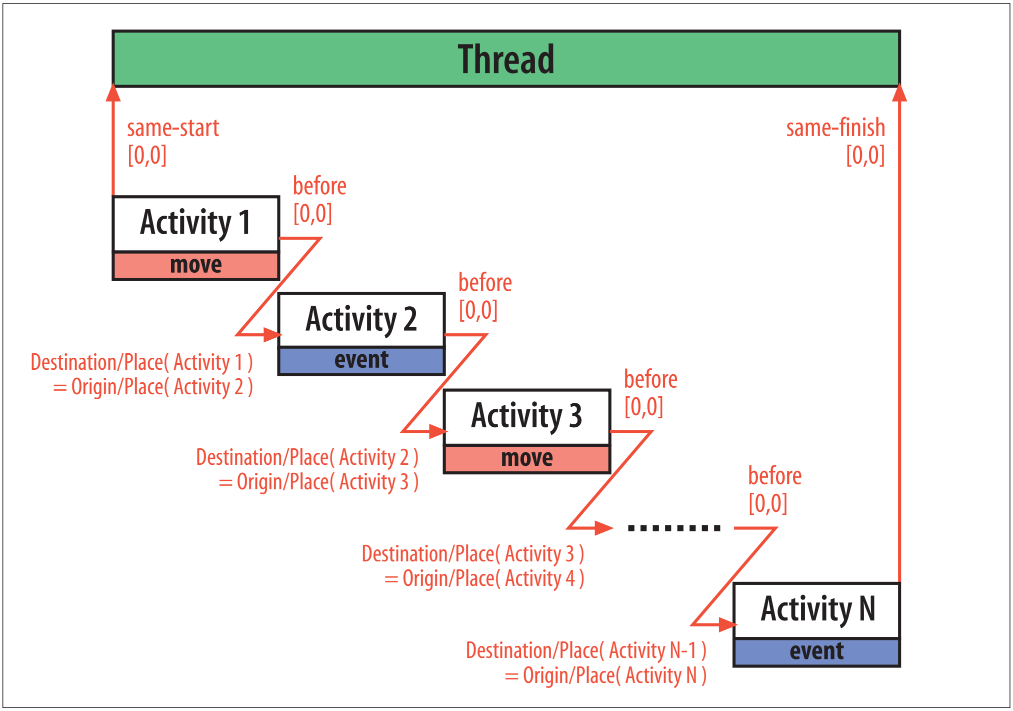

Thread
same-start
[0,0]
same-finish
[0,0]
Activity 1
move
before
[0,0]
Activity 2
event
before
[0,0]
Destination/Place(Activity 1)
= Origin/Place(Activity 2)
Activity 3
move
before
[0,0]
Destination/Place(Activity 2)
= Origin/Place(Activity 3)
before
[0,0]
Destination/Place(Activity 3)
= Origin/Place(Activity 4)
Activity N
event
Destination/Place(Activity N-1)
= Origin/Place(Activity N)

Figure 15: Thread structure

An example (albeit simplified) of the use of threads in a plan is provided in the Section 5.2, which describes the *Embassy Rescue* scenario.[14]

[13]A more detailed description of how resources are assigned to threads is presented in Section 3.2.2.6 (**Taskforces**), which introduces the concept of a *taskforce*.

[14]Threads must currently be defined using an additional data file that accompanies the *scenario*-`plan.xml` file,

3.2.2 Resources

Comirem supports the definition of both stationary and mobile renewable resources.[15] Stationary resources (called either *places* or *components*) reside at a fixed geographic location, while mobile resources may move—or be moved—from location to location at speeds determined by their characteristics and configuration. Usage of stationary resources (e.g., bases, airstrips) is a function of (typically) their availability and capability to accommodate various tasks. That is, stationary resources are often used simply as placeholders/anchors within a plan to determine movement requirements for mobile resources, and in such cases are often not actually allocated to activities (places and components are described further in Section 3.2.2.5). They can, however, be required by activities and allocated by the scheduler. Mobile resources, alternatively, are more complex. In some cases (e.g., involving transport vehicles), usage depends not only on their capability to support a given task, but also on such additional factors as capacity, speed, location and (historically) range.[16] In other cases (e.g., personnel), usage can occur at different locations but may require transport between locations. Resources can also be grouped into higher-level *aggregate* resources (e.g., a military taskforce), to perform activities as a single entity for some period of time.

All resources in Comirem are unit-capacity resources: that is, they can accommodate only a single activity at a time—unless defined as *uncapacitated* place components (see Section 3.2.2.5). Extension of the Comirem resource model to support multi-capacity resources has been identified as a logical future extension (see Section 6).

3.2.2.1 The Resource Requirement Hierarchy

Comirem provides a base group of common mobile resource types, such as `aircraft`, `seacraft` and `landcraft`, which define generic vehicle classes that may be further specialized as part of a particular domain model. Additionally, the `FOOT` resource type provides an *organic* mobile resource (i.e., of type `:organic`) that can be used to represent the movement of human resources. Accompanying these predefined resource types are a set of model types, such as `ROTOR` and `FIXED-WING` for aircraft, `WHEELED` and `TRACKED` for landcraft, and `POWER` and `ROW` for seacraft. Together with the set of capabilities, resource types and resource instances defined for a particular domain (i.e., a situation), model types provide a fourth level in the hierarchy for specifying resource requirements for activities at multiple levels of abstraction.

Figure 16 illustrates a subset of the resource-requirement hierarchy for aircraft in a particular domain. The `light-transport` capability can be achieved using the `MH-47` and `MH-60` resources classes, while the `air-drop` capability can be achieved using the `MH-60` and `MC-130H` resource classes. Both the `MH-47` and `MH-60` resources are aircraft of model type `ROTOR`, while the `MC-130H` is a `FIXED-WING`. Finally, three instances of `MH-60` are shown at the bottom: `MH-60-001`, `MH-60-002` and `MH-60-003`. As mentioned earlier, activities can specify resource requirements at any of these four levels of abstraction, which delivers valuable flexibility to the planner in cases where specific requirements are not appropriate.

since Comirem does not yet provide an interactive means nor an XML data file for thread specification. This is a temporary limitation.

[15]At the present time, Comirem does not support consumable resources (e.g., for items such as fuel, ammunition, money).

[16]For aircraft, range is becoming less of a constraint on usage, owing to the increased use of in-flight refueling.

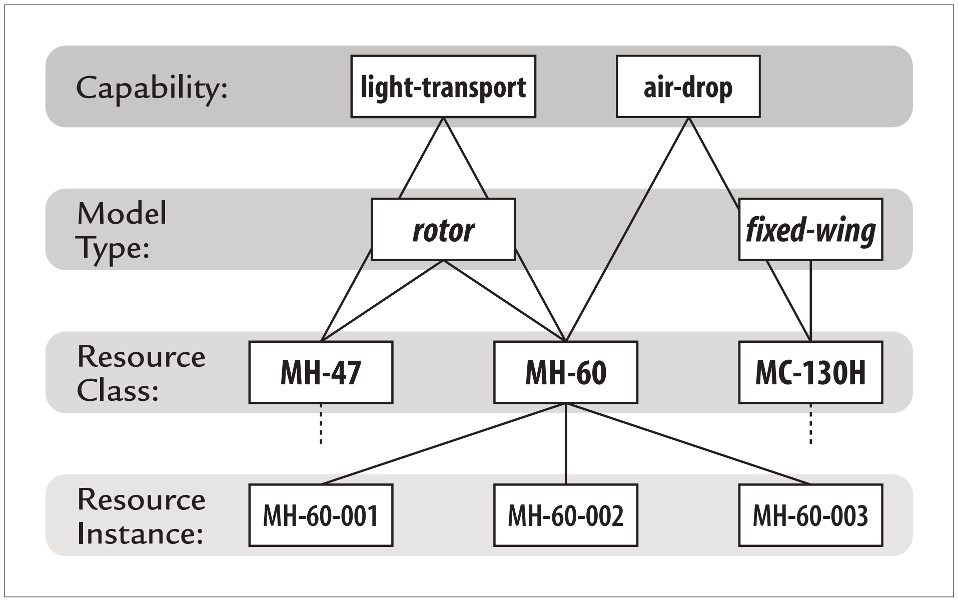


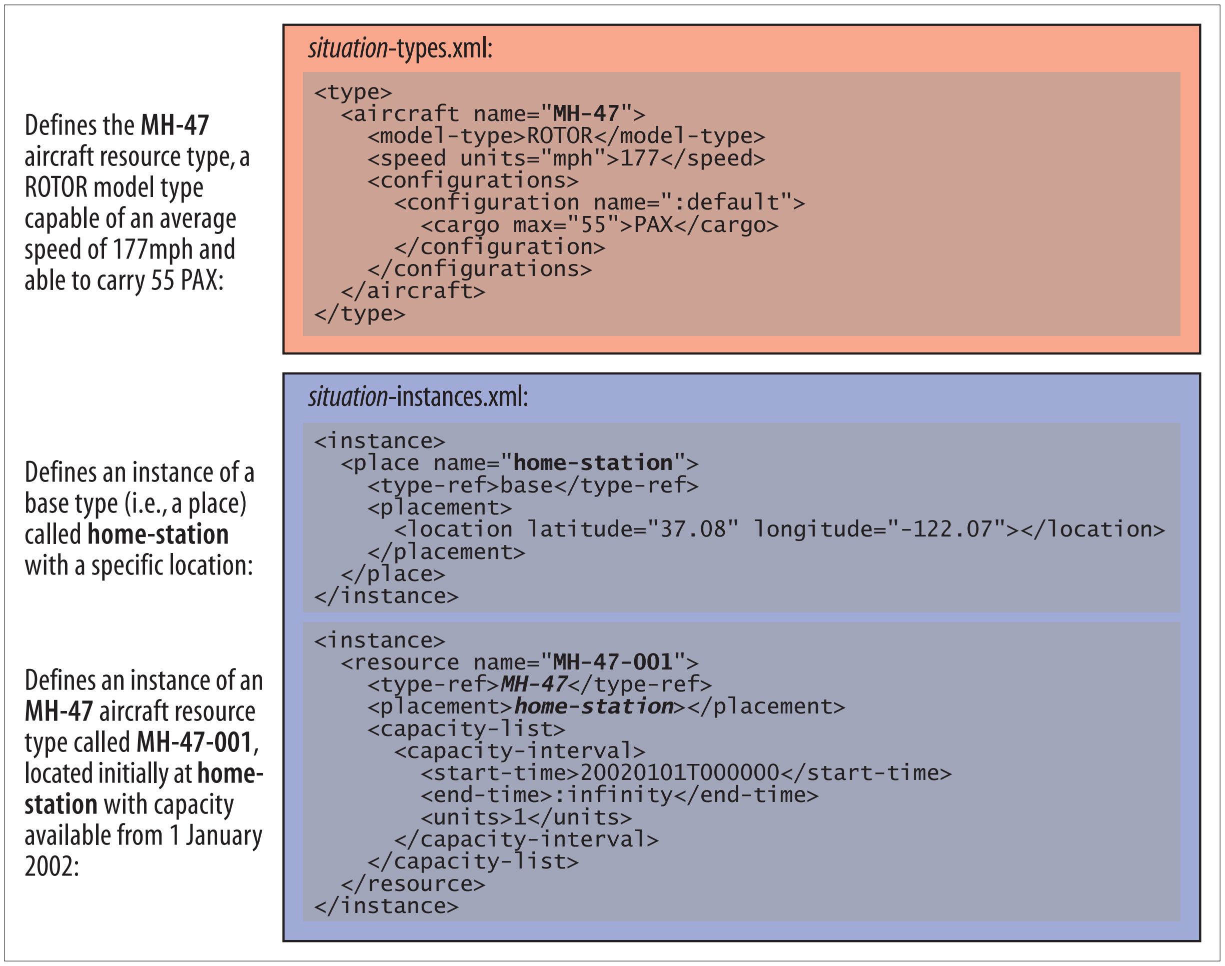
Figure 16: Subset of an aircraft-specific resource-requirement hierarchy

3.2.2.2 Domain-Specific Resource Types

It is not within the purview of Comirem to provide support for all conceivable resource classes. Toward that end, and akin to the similar support for the definition of domain-specific activity types described in Section 3.2.1.1 (**Hierarchical Activity Networks**), Comirem does provide a mechanism for users to define new domain-specific resource classes through the extension of existing Comirem classes (such as `aircraft`, `landcraft` and `seacraft`).

Figure 17 illustrates the definition of both a domain-specific resource type and an instance of this new type. An `MH-47` is defined as an aircraft (a helicopter of model type `ROTOR`) with an average air speed of 177 mph, which in its default configuration can accommodate as cargo, 55 `PAX` (i.e., personnel). Upon reading and processing the *situation*`-types.xml` file for this domain, Comirem will establish the `MH-47` resource type by generating and evaluating the appropriate class definition. Subsequent `MH-47` instance definitions appearing inside the *situation*-`instances.xml` file will lead to the creation of `MH-47` instances.

From the perspective of the Comirem scheduler, the salient features of a resource are: (1) its availability and (2) its influence on the expected duration of a requiring activity. For mobile resources, the expected duration for a move activity is determined by the specified origin and destination and the average speed of the resource, computed using a Great Circle calculation. At the present time, in all other cases (i.e., determining the expected durations for stationary resource reservations and non-moving mobile resource reservations), Comirem must rely on durations specified using duration constraints accompanying activity definitions. The resource attributes necessary for providing this information to the scheduler are identified in the following section. Just as there is no formal way to specialize behavior on user-defined, domain-specific activity types, Comirem presently lacks a formal mechanism for doing the same with resource types. This is a logical future extension that is discussed in Section 6.



Defines the **MH-47** aircraft resource type, a ROTOR model type capable of an average speed of 177mph and able to carry 55 PAX:

situation-types.xml:

```
<type>
  <aircraft name="MH-47">
    <model-type>ROTOR</model-type>
    <speed units="mph">177</speed>
    <configurations>
      <configuration name=":default">
        <cargo max="55">PAX</cargo>
      </configuration>
    </configurations>
  </aircraft>
</type>
```

Defines an instance of a base type (i.e., a place) called **home-station** with a specific location:

situation-instances.xml:

```
<instance>
  <place name="home-station">
    <type-ref>base</type-ref>
    <placement>
      <location latitude="37.08" longitude="-122.07"></location>
    </placement>
  </place>
</instance>
```

Defines an instance of an **MH-47** aircraft resource type called **MH-47-001**, located initially at **home-station** with capacity available from 1 January 2002:

```
<instance>
  <resource name="MH-47-001">
    <type-ref>MH-47</type-ref>
    <placement>home-station</placement>
    <capacity-list>
      <capacity-interval>
        <start-time>20020101T000000</start-time>
        <end-time>:infinity</end-time>
        <units>1</units>
      </capacity-interval>
    </capacity-list>
  </resource>
</instance>
```

Figure 17: Annotated XML specifications of resources

3.2.2.3 Resource Attributes

The attributes for resources, both stationary and mobile, are described in Tables 4 and 5, and follow the XML grammars described in Section A.3. As with the activity instances discussed in Section 3.2.1.2 (**Activity Attributes**), note that values for these attributes can be provided either at the class level, as part of the definition of a specialized resource type, or at the instance level, as part of the definition of a specific resource instance.

Table 4: Resource type attributes

- *Name* [**required**]

  The name of the resource type

- *Description* [optional]

  A description of the resource

- *Model-type* [optional]

  The name of a predefined model type (e.g., `ROTOR`, `FIXED-WING`), for classifying the resource

- *Configurations* [optional]

  A collection of *configurations* that describe the capabilities of a resource depending on how it is physically configured[a]

  Resource configurations are specified in terms of the aforementioned predefined cargo types (e.g., `PAX`, `MUNITIONS`, `FUEL`). The default configuration permits unlimited cargo capacity on a resource (i.e., no siblings are required when using it).

- *Capabilities* [optional]

  A list of capabilities that can be delivered by this resource

  Capabilities appearing in this list need not be predefined. If referenced for the first time in this list, a new capability object will be created on the fly and linked to the resource.

- *Speed* [**required**]

  For a *mobile* resource, its average traveling speed

- *Range* [*ignored*]

  For a *mobile* resource, its maximum travel range[b]

[a]Configurations in COMIREM are currently treated as though they can be changed instantaneously, as if they are achievable by means of zero-duration setup actions. Note, however, that the modifications necessary to support true situation-dependent setups are relatively minor.

[b]At the present time, COMIREM does not enforce range constraints.

Table 5: Resource instance attributes

- *Name* [**required**]

  The name of the resource

- *Type-Ref* [**required**]

  The type of the resource: either `mobile`, `place`, `component`, `aircraft`, `landcraft` or `seacraft`

- *Placement* [**required**]

  The placement of a resource

  For mobile resources this is an *initial* location.[a]

  Places can only be placed at latitude/longitude locations. Mobile resources and components can be placed at either a latitude/longitude location or another place instance.

- *Components* [optional]

  For *place* resources, the collection of capacitated component resources residing with them (e.g., runways at an airport)

- *Uncapacitated-components* [optional]

  For *place* resources, the collection of unlimited-capacity component resources residing with them, specified in terms of capability, model type or resource class

- *Capacity-list* [optional]

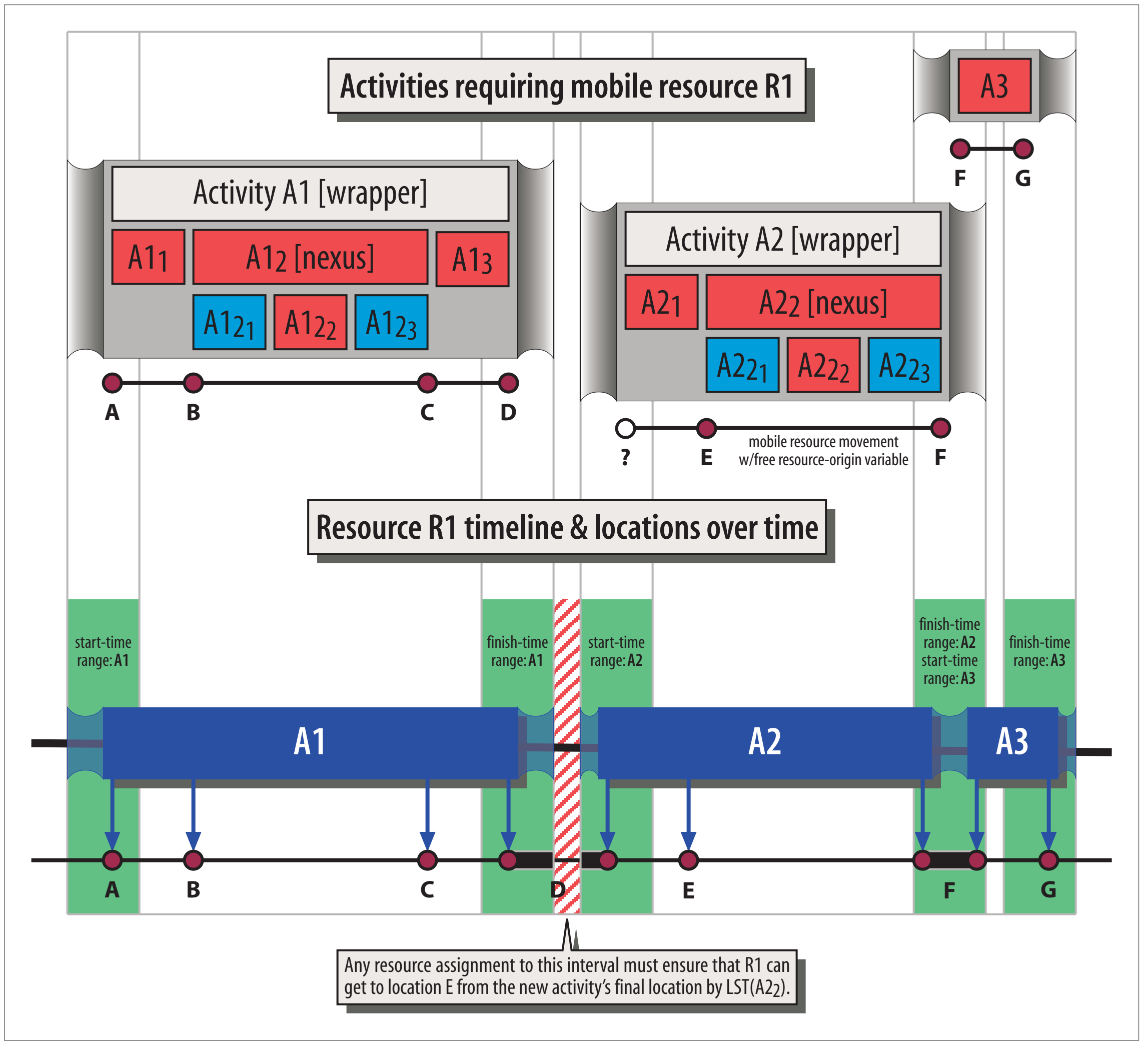
  A sequence of possibly non-contiguous capacity intervals, indicating at the outset, the time windows during which the resource is available for allocation (i.e., akin to business hours)

  Note that COMIREM currently only supports unitary capacity intervals. The default capacity window extends for the entire scheduling horizon.

[a]An altitude can also be provided, although it is currently ignored. Travel-distance calculations are made under the assumption that all locations are at the same altitude (e.g., sea level).

3.2.2.4 Flexible Timelines

As has been mentioned earlier, Comirem implements a *flexible time-bounds* scheduling approach that avoids assigning specific fixed time intervals to the activities in a plan when allocating resources. Instead, resource assignments become floating reservations that can move forward and backward (i.e., downstream and upstream) in time according to their latest finish times and earliest start times, respectively. As activities are scheduled on a resource, existing reservations are pushed aside to make room for the insertion of new reservations. It is only when such shifting will not open up enough room for a new reservation that the scheduler gives up and considers another resource candidate instead.



Activities requiring mobile resource R1
A3
F
G
Activity A1 [wrapper]
A1₁
A1₂ [nexus]
A1₃
A1₂₁
A1₂₂
A1₂₃
A
B
C
D
Activity A2 [wrapper]
A2₁
A2₂ [nexus]
A2₂₁
A2₂₂
A2₂₃
?
E
mobile resource movement w/free resource-origin variable
F
Resource R1 timeline & locations over time
start-time range: A1
finish-time range: A1
start-time range: A2
finish-time range: A2 start-time range: A3
finish-time range: A3
A1
A2
A3
A
B
C
D
E
F
G
Any resource assignment to this interval must ensure that R1 can get to location E from the new activity's final location by LST(A2₂).

Figure 18: Insertion of activities onto a mobile resource timeline

In Figure 18 (top), note that three activities: **A1**, **A2** and **A3**, all require the use of resource **R1**. Note also that activity **A1** requires **R1** to be sourced from location **A** and returned to location **D**, while **A2** has its resource origin declared as a free variable and leaves **R1** at its destination location **F**. Activity **A3** simply requires **R1** to start at location **F** and finish up at **G**. Activities **A2** and **A3** can easily be scheduled in sequence on **R1** because of the shared resource location **F** and the flexibility in the finish-time range for **A2** and the start-time range for **A3** (i.e., despite the overlap, there is still room for both to use **R1**). Similarly, activities **A1** and **A2** have no temporal overlap at all, even if **A2** has to secure **R1** from location **D**.

Once activities **A1**, **A2** and **A3** have been assigned to resource **R1** (bottom of Figure 18), the interval of available capacity between **A1** and **A2** takes on an additional (implicit) constraint, which is that any reservation within that interval must leave activity **A2** with enough time for it to source **R1** from the new activity's final location to location **E** by the latest start time of $\mathbf{A2_2}$. Otherwise, until either of the reservations for **A1** or **A2** is modified or retracted, the new activity cannot be inserted into that interval.

3.2.2.5 Place Components

A stationary *place* resource in COMIREM can serve as its own resource and also provide a collection of component resources that reside at its location. A typical example is an airport, which can provide runways and aircraft-parking areas that can be reserved individually. Such components can also be *uncapacitated*, in which case they provide unlimited capacity—at a particular location—specified in terms of capability, resource type or model type. Components are allocated to *events* that require their capacity and reference their corresponding place (i.e., component capacity at place **P** can only be allocated to events with a `<place>`**P**`</place>` tag).

Figure 19 presents a collection of XML forms that demonstrate: (1) the definition of a place-component resource type called `working-MOG`, (2) a capacitated instance of `working-MOG` called `embassy-yard` located at an `embassy` place instance, (3) the `embassy` place instance showing `embassy-yard` as a component, (4) a place instance called `home-station` providing unlimited (i.e., uncapacitated) `working-MOG` capacity, and finally, (5) an activity type called `load-step` that requires `working-MOG` capacity. Simply put, this example describes two sources of `working-MOG` capacity that may be used by instances of the `load-step` activity type. If `working-MOG` is required from the `embassy` location, it must come from the `embassy-yard` place component and be allocated by the scheduler depending on its availability. If, on the other hand, `working-MOG` is required from the `home-station` location, it is automatically available without any interaction with the scheduler, because its capacity is unlimited.

Places can be defined as part of either a situation (i.e., within the *situation*-`instances.xml` data file) or a scenario (i.e., within the *scenario*-`plan.xml` data file). The distinction is subtle but relevant. The determining factor is whether the place is conceptually a part of the bed-down information that comprises a situation or is a location specific to a particular scenario. For example, a default bed-down situation may define a collection of military aircraft wings deployed among a number of bases to be used by a range of scenarios. In this case, the scenario must conform to the constraints dictated by the geographic placement of the resources defined in the situation. In the latter case, places are locations that are specific to the scenario, in that they define points in the plan where actions are expected to occur, and can be legitimately referenced within the plan. In most cases, it is to be expected that some places will be defined as part of the situation while others are specific to, and defined as part of, a particular scenario.

① Defines the **working-MOG** place-component resource type, giving it a maximum serving capacity of a single AIRCRAFT:

```
<type>
  <component name="working-MOG">
    <configurations>
      <configuration name=":default">
        <cargo max="1">AIRCRAFT</cargo>
      </configuration>
    </configurations>
  </component>
</type>
```

② Defines an instance of a **working-MOG** place-component called **embassy-yard** to be located at an **embassy** place instance (defined below), with unitary capacity and unlimited availability:

```
<instance>
  <component name="embassy-yard">
    <type-ref>working-MOG</type-ref>
    <placement>embassy</placement>
    <capacity-list>
      <capacity-interval
        <start-time>20010101T000000</start-time>
        <end-time>:infinity</end-time>
        <units>1</units>
      </capacity-interval>
    </capacity-list>
  </component>
</instance>
```

③ Defines the **embassy** place instance listing the **embassy-yard** as a (capacitated) place-component:

```
<instance>
  <place name="embassy">
    <type-ref>base</type-ref>
    <placement>
      <location latitude="38.17" longitude="-122.3"></location>
    </placement>
    <components>embassy-yard</components>
  </place>
</instance>
```

④ Defines the **home-station** place instance providing uncapacitated **working-MOG** capacity:

```
<instance>
  <place name="home-station">
    <type-ref>base</type-ref>
    <placement>
      <location latitude="37.08" longitude="-122.07"></location>
    </placement>
    <uncapacitated-components>
      <resource-class>working-MOG</resource-class>
    </uncapacitated-components>
  </place>
</instance>
```

⑤ Defines the **load-step** (event) activity type requiring **working-MOG** resource capacity:

```
<type>
  <event name="load-step">
    <requirements>
      <requirement>
        <resource-class>working-MOG</resource-class>
      </requirement>
    </requirements>
  </event>
</type>
```

Figure 19: Annotated XML specifications of (un)capacitated places and place components

3.2.2.6 Taskforces

Somewhat analogously to the way in which multiple activities can be grouped together into threads (as described in Section 3.2.1.5, **Threads**), COMIREM also supports the aggregation of resources into high-level *taskforces*. A taskforce is a collection of resources that have been grouped together for a specific period of time, during which they can be allocated collectively—to threads—as a single resource object. Taskforce resources are required to be either mobile resources or *movable* (i.e., non-place-component) stationary resources that can be transported, since they must be able to move, or be moved, from location to location as required by a thread. Certain predefined mobile resource classes (e.g., those that are typically sourced from base locations) may require that additional positioning and depositioning legs be instantiated to deliver them to the starting location of a thread and back from the ending location. Mobile resources that are not positionable are required to already be located at the start location for the thread to which the taskforce will be assigned.

The resource-aggregation capability provides two key benefits:

1. *Scheduling efficiency through aggregation:*

   The definition of a taskforce facilitates the allocation of a single collection of resources to perform a sequence of activities comprising a thread, so that COMIREM need not allocate each individual resource to the thread.

2. *Explicit and detailed tracking of constituent resources:*

   Constituent resources within a taskforce can be easily located over time as threads are scheduled and executed. Since taskforces are treated as single objects, their specific make-up can be ignored at the scheduling level. But their make-up *is* recorded explicitly, and can therefore be used to rapidly determine both *where a specific constituent resource is located*, and *which constituent resources are present at a location*, at any point in time. For real-time battle management, these are important queries to be capable of answering.

The duration of a taskforce depends on the thread to which it is allocated.[17] Until it is allocated to a thread, a taskforce's duration is unbounded. Throughout the duration of a taskforce, its constituent resources may not be allocated to any other activity or taskforce, in which case, if a taskforce has not yet been allocated to a thread, its constituents can *only* be allocated to that taskforce. Once a taskforce's bounds have been set by its allocation to a thread, its constituent resources are free to be allocated to other activities and taskforces whose temporal bounds do not overlap those of the taskforce to which they are already assigned.

Constituent resources are available for use by any and all activities within the thread to which they are allocated. Since the constituents are required to be mobile in nature, they are assumed to be able to travel from location to location as dictated by the thread's activity sequence. And since a thread's activity sequence is precluded from allowing overlap, there is no possibility of two activities within a thread attempting to use the same resource at the same time.

Figure 20 illustrates the relationships between a taskforce, its constituent resources, and a thread. Like a typical COMIREM resource, each taskforce maintains its own timeline upon which

[17]Currently, COMIREM restricts taskforces from being assigned to more than a single thread. One way around this restriction, however, is to define multiple taskforces, assign each to a desired thread, and then allocate the same set of constituent resources to each taskforce.

its thread assignment is placed, like a regular resource reservation. When a resource joins a taskforce, a *taskforce-participation-event* is placed on its timeline, with its duration and flexibility dictated by the duration and flexibility of a thread possibly assigned to the taskforce (via multiple `same-start` and `same-finish` temporal constraints), and whether or not derivative (de)positioning legs are required. If no thread has been assigned to the taskforce, the duration of the *taskforce-participation-event* is unbounded, and there is no flexibility (i.e., the *taskforce-participation-event* reservation takes up the resource's entire timeline).

The overall feasibility of a taskforce assignment (to a thread) is dependent on the timelines of its constituent resources and the requirements of the thread to which it may be assigned. Additionally, the existing timelines of the constituent resources assigned to a taskforce constrain the extent of both the taskforce and the thread to which it may be assigned.

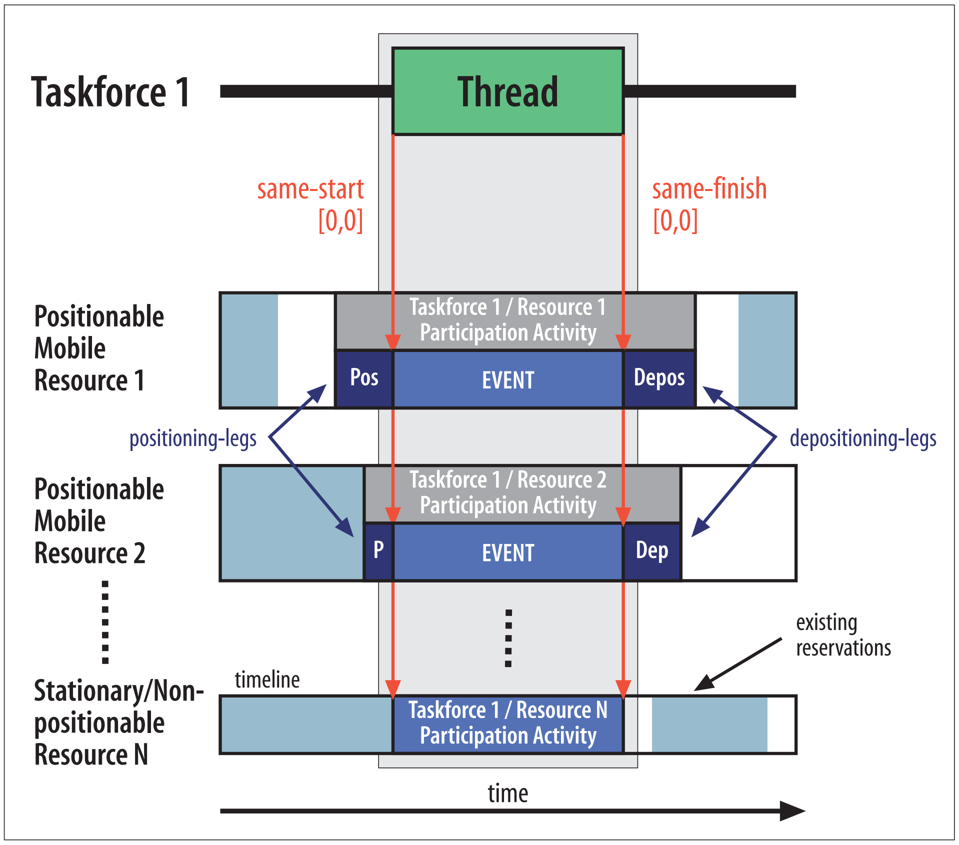

Taskforce 1
Thread
same-start [0,0]
same-finish [0,0]
Positionable Mobile Resource 1
Taskforce 1 / Resource 1 Participation Activity
Pos
EVENT
Depos
positioning-legs
depositioning-legs
Positionable Mobile Resource 2
Taskforce 1 / Resource 2 Participation Activity
P
EVENT
Dep
existing reservations
timeline
Stationary/Non-positionable Resource N
Taskforce 1 / Resource N Participation Activity
time

Figure 20: Taskforce structure and relationships to thread and resource timelines

It should be noted here that currently there are some limitations on the means by which taskforces are defined in COMIREM. Specifically, taskforces must be assembled by hand using the **Resource & Taskforce Manager** in the COMIREM GUI (described in Section B.4.2).[18]

An example of the use of threads in COMIREM can be found in the description of the `Embassy-Rescue` domain presented in Section 5.2, in which three taskforces are defined to undertake

[18]Augmentation of the COMIREM data file XML formats (described in Appendix A) is anticipated to allow taskforces to be specified declaratively as part of an input situation.

three different threads comprising a mission to rescue personnel from a United States embassy on foreign soil.

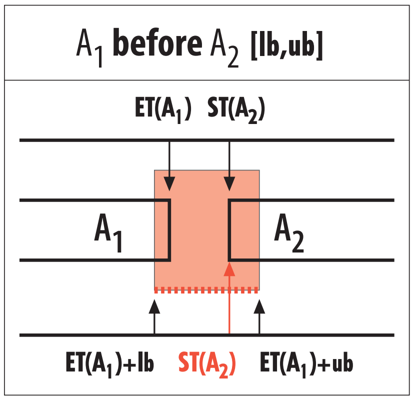
3.2.3 Constraints

In addition to the previously mentioned constraints on activities and resources, COMIREM provides the ability to synchronize activities (and by extension, threads) using three different kinds of constraints. A set of temporal sequencing constraints governs the relationships between activities, a reference-hour constraint facilitates the linking of activities to movable time points, and an anchoring constraint facilitates the placement of lower and upper bounds on the start and finish times of activities. These constraint types are described in the following subsections.

3.2.3.1 Temporal Sequencing Constraints

To support the relative sequencing of activities, COMIREM implements a set of canonical binary temporal relations, such as *before*, *same-start* and *same-finish*, and *overlaps* and *contains*. These relations enforce sequencing rules while attempting to preserve flexibility within the time bounds of constrained activities.

Figure 21 illustrates the *before* constraint (also called *successor*), which can also be used to implement an *after* relation, by reversing the order of the activities. The *before* constraint, coupled with a [lb,ub] temporal range, forces one activity (i.e., $\mathbf{A_2}$) to start within a time window offset—shown in red—from the end time of another activity (i.e., $\mathbf{A_1}$). Note that if the lower bound is negative, then overlap between the activities is allowed.



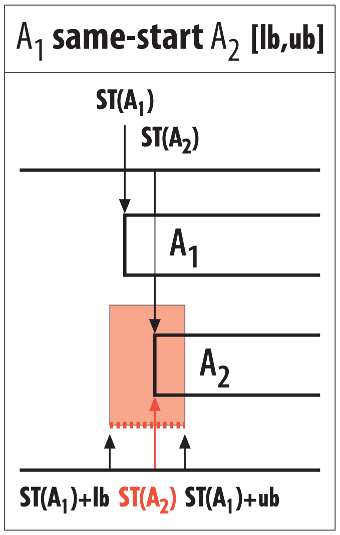
before(A_1, A_2 lb, ub)

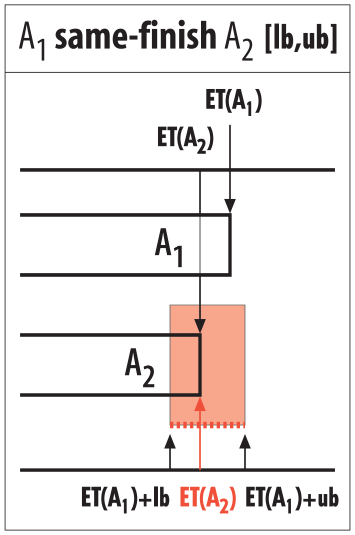
$[ET(A_1) + lb] \leq ST(A_2) \leq [ET(A_1) + ub]$

default values: [lb,ub]: $[0,\infty]$

Figure 21: *Successor* (also *before* and *after*) constraint

In Figure 22, the *same-start* and *same-finish* constraints are illustrated. In the left half of the figure, the *same-start* relation forces one activity (i.e., $\mathbf{A_2}$) to start within a time window offset—again shown in red—from the start time of another activity (i.e., $\mathbf{A_1}$). In the right half of the figure, the *same-finish* relation forces one activity (i.e., $\mathbf{A_2}$) to finish within a time window offset from the end time of another activity (i.e., $\mathbf{A_1}$). Note that a negative lower bound on the *same-start* constraint permits $\mathbf{A_2}$ to actually start before $\mathbf{A_1}$, and that a negative upper bound on the *same-finish* constraint permits $\mathbf{A_2}$ to actually finish after $\mathbf{A_1}$.

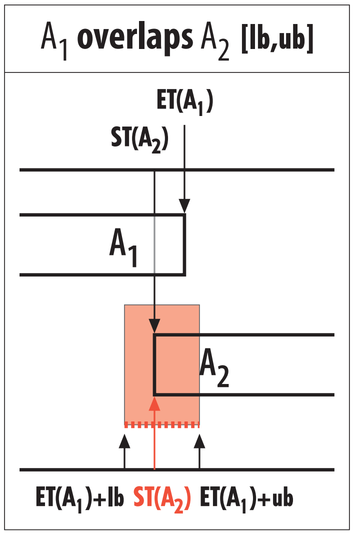


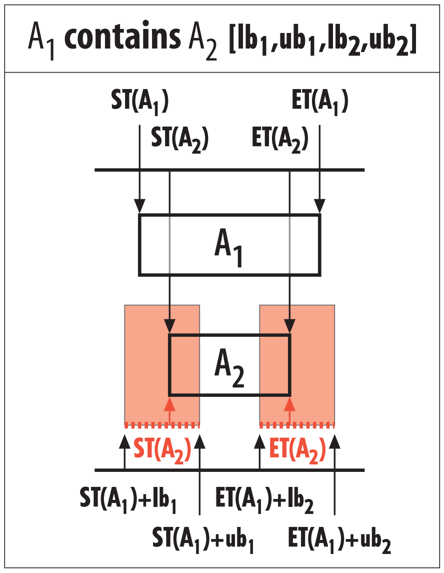


same-start(A_1, A_2, lb, ub)	**same-finish**(A_1, A_2 lb, ub)
$[ST(A_1) + lb] \leq ST(A_2) \leq [ST(A_1) + ub]$	$[ET(A_1) + lb] \leq ET(A_2) \leq [ET(A_1) + ub]$
default values: [lb,ub]: $[0,\infty]$	*default values*: [lb,ub]: $[-\infty,0]$

Figure 22: *Same-start* and *same-finish* constraints

Figure 23 illustrates the *overlaps* and *contains* constraints. In the left half of the figure, the *overlaps* relation forces one activity (i.e., **A$_2$**), to start within a time window offset from the end time of another activity (i.e., **A$_1$**). In the right half of the figure, the *contains* relation forces one activity (i.e., **A$_2$**), to both (1) start within a time window offset from the start time, and (2) end within a time window offset from the end time, of another activity (i.e., **A$_1$**).





overlaps(A$_1$, A$_2$, lb, ub)	**contains**(A$_1$, A$_2$ lb$_1$, ub$_1$, lb$_2$, ub$_2$)
$[ET(A_1) + lb] \leq ST(A_2) \leq [ET(A_1) + ub]$ *default values*: [lb,ub]: $[-\infty,0]$	$[ST(A_1) + lb_1] \leq ST(A_2) \leq [ST(A_1) + ub_1]$ $\wedge\ [ET(A_1) + lb_2] \leq ET(A_2) \leq [ET(A_1) + ub_2]$ *default values*: [lb$_1$,ub$_1$]: $[0,\infty]$ [lb$_2$,ub$_2$]: $[-\infty,0]$

Figure 23: *Overlaps* and *contains* constraints

In all of these cases, the default [lb,ub] values are set to enforce a strict interpretation of the constraint relation. As has been pointed out, however, these relations can be relaxed through the specification of different lower and upper bounds.

The basic attributes of the five temporal sequencing constraint types are described in Table 6. The XML format for specifying instances of these constraints as part of a *scenario*-`plan.xml` data file is presented in Section A.1.4.1.

Table 6: Temporal sequencing constraint attributes

- *Type-Ref* [**required**]

  Indicates the type of constraint: must be one of `successor`, `before`, `same-start`, `same-finish`, `overlaps` or `contains`.

- *From* | *From-Type* [**required**]

  Identifies the set of activities from which the constraint originates (corresponding to the $\mathbf{A_1}$'s in Figures 21, 22 and 23)

  Notice that activities can be identified by name (using the *From* attribute), or by type (using *From-Type*). The latter is used primarily for specifying sequencing relations among child activities, in which case the types of the children must be unique.

- *To* | *To-Type* [**required**]

  Identifies the set of activities to which the constraint is directed (corresponding to the $\mathbf{A_2}$'s in Figures 21, 22 and 23)

  To-Type is used analogously to *From-Type*.

- *LB* [optional]

  Provides the lower bound on the time window for a constraint

- *UB* [optional]

  Provides the upper bound on the time window for a constraint

3.2.3.2 Reference Hour Constraints

To achieve a possibly tighter degree of synchronization of activities, Comirem provides *reference hour* (or *ref-hour*) constraints for linking sets of activities to a relative date, which can be assigned to a particular point in time and shifted as necessary. An activity can be linked to a ref-hour by either its start or finish time, or both. These links define what we call the *preferred* start and finish times for the activity.

One benefit of the reference hour constraint type is the ability to define collections of activities (i.e., plan fragments) whose temporal bounds can be manipulated collectively by shifting a single reference point in time. This capability is ideal for defining the *N hour* and *H hour* synchronization points typically used in military plans. During the planning/scheduling process, if

H hour is shifted by a day, then the preferred start/finish times of all activities linked to *H hour* will similarly be shifted by the same amount.[19]

Another benefit of the ref-hour constraint derives from the fact that each reference hour maintains its own flexible time window (defined by upper and lower-bound offsets from the ref-hour's time point), which establishes a "level of precision" with respect to the synchronization of activities (the default is ± 3 minutes). If the *preferred* start/finish times of linked activities occur within this level of precision (or *tolerance*) as execution unfolds, a plan is considered to be executing as planned.

The relationship between a reference hour, calendar zero, and a linked activity is illustrated in Figure 24.

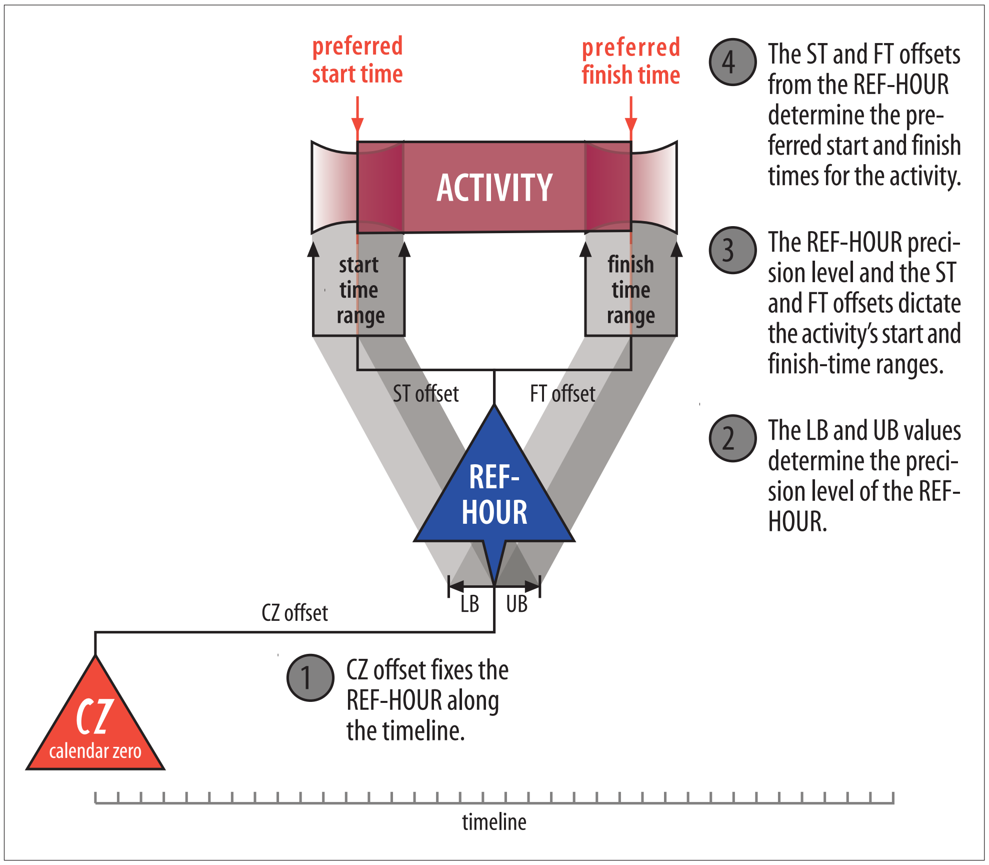

preferred start time
preferred finish time
ACTIVITY
start time range
finish time range
ST offset
FT offset
REF-HOUR
LB
UB
CZ offset
CZ
calendar zero
timeline
4 The ST and FT offsets from the REF-HOUR determine the preferred start and finish times for the activity.
3 The REF-HOUR precision level and the ST and FT offsets dictate the activity's start and finish-time ranges.
2 The LB and UB values determine the precision level of the REF-HOUR.
1 CZ offset fixes the REF-HOUR along the timeline.

Figure 24: Reference hour constraint representation

[19]Note that the flexibility of a reference hour is collectively constrained by its linked activities, so that it may only be shifted as long as sufficient flexibilityfor those activities exists in the schedule.

3.2.3.3 Anchoring Constraints

Anchoring constraints introduce lower and upper bounds on the start and finish times of activities, for defining *not-earlier-than* (i.e., NET or EST/EFT) and *not-later-than* (NLT or LST/LFT) temporal relations. Release dates and due dates are typical anchors: a release date imposes an EST constraint on the initial activities in a plan, while a due date imposes an LFT constraint on the final activities in a plan. Anchoring constraints are specified for activity instances using the EST, LST, EFT and LFT attributes described in Table 2. The anchoring constraint representation is illustrated in Figure 25.

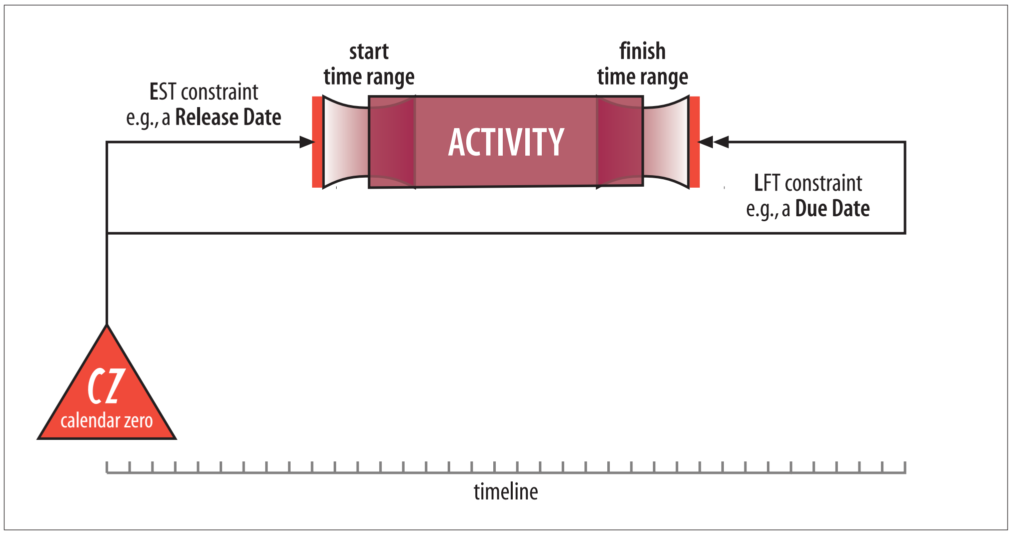


Figure 25: Anchoring constraint representation

3.2.4 Capabilities

As mentioned in Section 3.2 (**Conceptual Model**), Comirem supports both the direct mapping of activities to their required resources and the indirect mapping via *capabilities*, which collectively define a high-level categorization of resource types according to the kinds of services they provide (e.g., heavy and light transport, airlift, close air support).

Capabilities are defined in a *situation*-`capabilities.xml` input data file using the attributes and tags described in Table 7. Figure 26 provides an example of some standard SOF-related capabilities, which follow the XML grammar described in Section A.4.

Table 7: Capability attributes

- *Name* **[required]**

  The name of the capability, for reference in the resource-requirements of activities and the configurations of resources

- *Capable-Types* **[required]**

  The set of resource-classes that can deliver the capability

  The mapping between capabilities and resources need only be specified in one direction (i.e., from capability to resource using this tag, or from resource to capability using the `capabilities` or `configurations` tags; COMIREM will fill in the missing inverse links.

- *Type-Ref* **[required]**

  Characterizes the capability as applying to either a `move` or `event` activity

  This tag is primarily for helping to organize and present information to the user in a meaningful fashion - i.e., so that capabilities related to movement are only offered as options for `move` activities, and capabilities related to stationary actions are only offered as options for `event` activities.

```
<capability name="heavy-transport">
  <capable-types>MC-130H C-141</capable-types>
  <type-ref>move</type-ref>
</capability>

<capability name="light-transport">
  <capable-types>MH-60 MH-47</capable-types>
  <type-ref>move</type-ref>
</capability>

<capability name="air-drop">
  <capable-types>MH-60 MC-130H</capable-types>
  <type-ref>move</type-ref>
</capability>
```

Figure 26: XML specifications for capabilities

3.2.5 Plans

A plan in COMIREM is a collection of activities defining a single, coordinated mission. The *scenario*-`plan.xml` file in which a plan is defined contains the activity and constraint instances that describe the actions needing to be taken and the order in which they must be performed. Once loaded into an existing situation, the feasibility of a plan/scenario can be ascertained through a process of either automatic or manual interactive scheduling.

The last remaining data fragment used in specifying a plan is the *plan configuration*, which provides the necessary temporal information for interpreting the *scenario*-`plan.xml` data file. The various attributes for the `configuration` tag are described in Table 8 and follow the XML grammar described in Section A.5.

Table 8: Plan-configuration attributes

- *Display-Name* [optional]

  An appropriate display name for the plan (i.e., for use in the GUI)

- *Calendar-Zero* [optional]

  The base time value for all times specified within a plan

  This value is relevant whenever the times in a plan are specified as integer values, in which case each such value represents an absolute time (i.e., *not* an offset). (It is preferable to specify times consistently within a plan, i.e., either as integer values or explicit dates/times.) *The default is 0.*

- *Calendar-Zero-Date* [optional]

  A calendar date/time for initializing COMIREM's *calendar-zero* value

  Integer time values within a plan are mapped into COMIREM's internal temporal representation, which maintains its own *calendar-zero*.

  The default is the value of COMIREM's `(time-today)` *function*.

- *Ref-Hours* [optional]

  Specifications of all ref-hours for a plan

  Ref-hour specifications include the name of the ref-hour, its time point, and its delta value, which is used to provide a window of temporal flexibility around the time point.

- *Plan-Horizon* [optional]

  The horizon for the plan *The default is 24 hours.*

- *Time-Granularity* [optional]

  The temporal granularity of the plan *The default is 'seconds.'*

3.3 Scheduler

COMIREM takes a *flexible-times* approach to allocating resources to activities over time. Activities in the plan are allowed to float within the time bounds that are permitted by current problem constraints, and the feasibility of various resource assignments is ensured by sequencing any pair of activities that are assigned the same resource. This approach can be contrasted with most current scheduling tools, which ensure resource feasibility by assigning *fixed* start and end times when allocating resources to activities. Fixed-time scheduling can simplify the enforcement of various resource usage constraints but it also generally implies considerable overcommitment. Under a flexible-times scheduling approach, alternatively, greater flexibility is retained. At both plan development and plan execution stages, this minimizes the need for change.

3.3.1 Managing Temporal Constraints

Within this flexible-times framework, temporal constraint propagation and consistency enforcement is achieved by encoding various elements of the plan as a Simple Temporal Problem (STP) constraint network [Dechter *et al.*, 1991], and applying an incremental STP constraint network solver. Briefly, temporal constraints are represented in an underlying graph $G\langle N, E\rangle$, where nodes in N represent time points, and edges in E are distances (labeled as [*lower bound, upper bound*] pairs) between the time points in N. A special time point, called "calendar zero," grounds the network and has the value 0. The network maintains lower and upper bounds on the time points by propagating the bounds on the distances of the edges. Activities, ref-hours, anchors and temporal sequencing constraints are uniformly represented as temporal constraints (i.e., edges) between relevant start and finish time points, as depicted in Figure 27. Planning/scheduling decisions generally correspond to the introduction of new constraints into the network (e.g., sequencing two activities that have been assigned the same resource) or the adjustment of existing constraints (e.g., refining the duration of an activity, moving an anchor). In either case, constraint propagation updates the bounds of affected nodes and checks for cycles in the resulting network. The lack of any such cycle ensures continued temporal feasibility of the plan. Otherwise a conflict has been detected, and backtracking (or some amount of constraint relaxation) is necessary.

3.3.2 Managing Resource Constraints

As implied above, basic resource feasibility is maintained by enforcing a disjunctive constraint on the execution of any two activities utilizing the same resource. If two such activities would otherwise have potentially overlapping execution windows, preservation of feasibility requires that an explicit sequencing constraint be introduced. Note that for an activity requiring several resources, this can imply several distinct links.

COMIREM provides a number of more specialized resource constraint management techniques, aimed at satisfaction of additional resource usage constraints:

- *Resource Location*
  To support a given move activity, a resource must be at the activity's origin location at its start and will be at the destination location at its finish. Depending on the location of the resource prior to its allocation to this activity, a supporting positioning activity might be

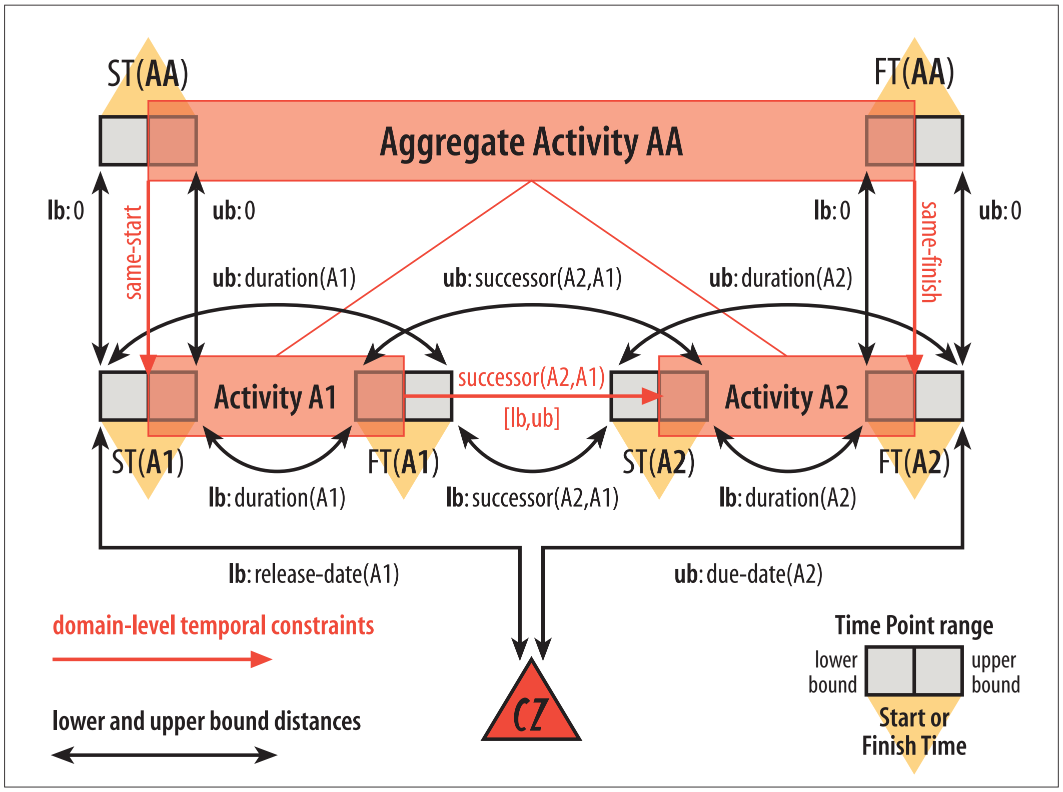


Figure 27: STP constraint network fragment with superimposed activities

necessary and likewise, depending on its subsequent obligations, depositioning might be necessary. As discussed earlier, a move activity expansion typically includes provisions for such *derivative* activities. As a given resource is assigned, location constraints are enforced by appropriately adjusting relevant duration constraints.

- *Resource Carrying Capacity*
  A given move activity specifies a manifest quantity which, together with the carrying capacity of available resources, will ultimately determine how many trips will be required and how long the overall move will take. When resources of a given type are allocated to a given activity, *sibling* activity networks are created and instantiated to correspond with each required trip. Depending on the number of resources actually available, these trips may occur in multiple waves, with some resources being reused.

3.4 Mixed-Initiative Resource Management

The above constraint management and scheduling procedures are configured to provide a range of interactive and semi-automated resource management capabilities within Comirem. Generally speaking, Comirem is designed to provide a graphical spreadsheet-style model of user-system interaction, where the user opportunistically manipulates various planning and plan data (e.g., requirements and capabilities, resource allocation decisions) and the consequences of user changes (e.g., updated sets of decision options, instantiation of implied derivative activity networks) are made apparent. The user can selectively employ more automated capabilities (e.g., requesting resource feasibility checking and auto-scheduling for larger fragments of the developing plan), in which case the resulting "propagation of effects" of a user action can be significant (and may involve heuristic choice). Also following the spreadsheet metaphor, the user can change her mind at any point, retract one or more prior actions, and resume the planning process in another direction.

The current planning focus in Comirem is on the problem of assigning resources to a (possibly evolving) plan and then managing these assignments as execution of the plan moves forward. Input to the system takes the form of an initial *plan sketch* that specifies, at some level of abstraction, the actions necessary to accomplish certain end goals for a given scenario. Figure 28 gives an example of the type of problem of interest. It graphically depicts a plan for evacuating citizens from an embassy in a foreign country. The same plan is shown from an activity-centered perspective in Figure 29. Following an initial staging activity, the `Embassy-Rescue` plan consists

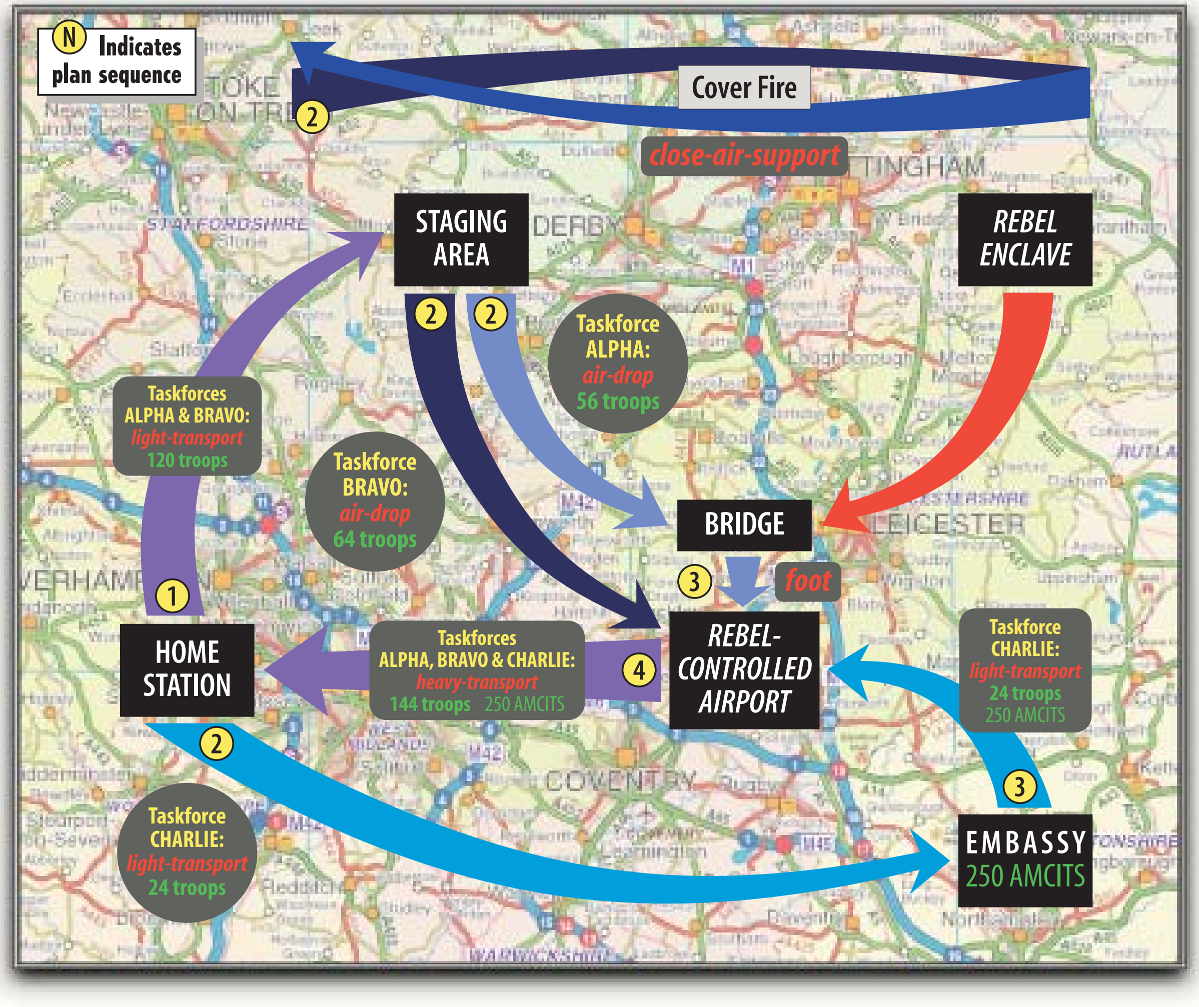

N Indicates plan sequence
Cover Fire
close-air-support
STAGING AREA
REBEL ENCLAVE
Taskforce ALPHA: air-drop 56 troops
Taskforces ALPHA & BRAVO: light-transport 120 troops
Taskforce BRAVO: air-drop 64 troops
BRIDGE
foot
REBEL-CONTROLLED AIRPORT
HOME STATION
Taskforces ALPHA, BRAVO & CHARLIE: heavy-transport 144 troops 250 AMCITS
Taskforce CHARLIE: light-transport 24 troops 250 AMCITS
Taskforce CHARLIE: light-transport 24 troops
EMBASSY 250 AMCITS
1
2
3
4

Figure 28: Embassy rescue scenario

basically of three parallel threads: one aimed at securing a local airport for eventual extraction (`Seize-Airport`), one concerned with blocking the advance of hostile forces to the airport

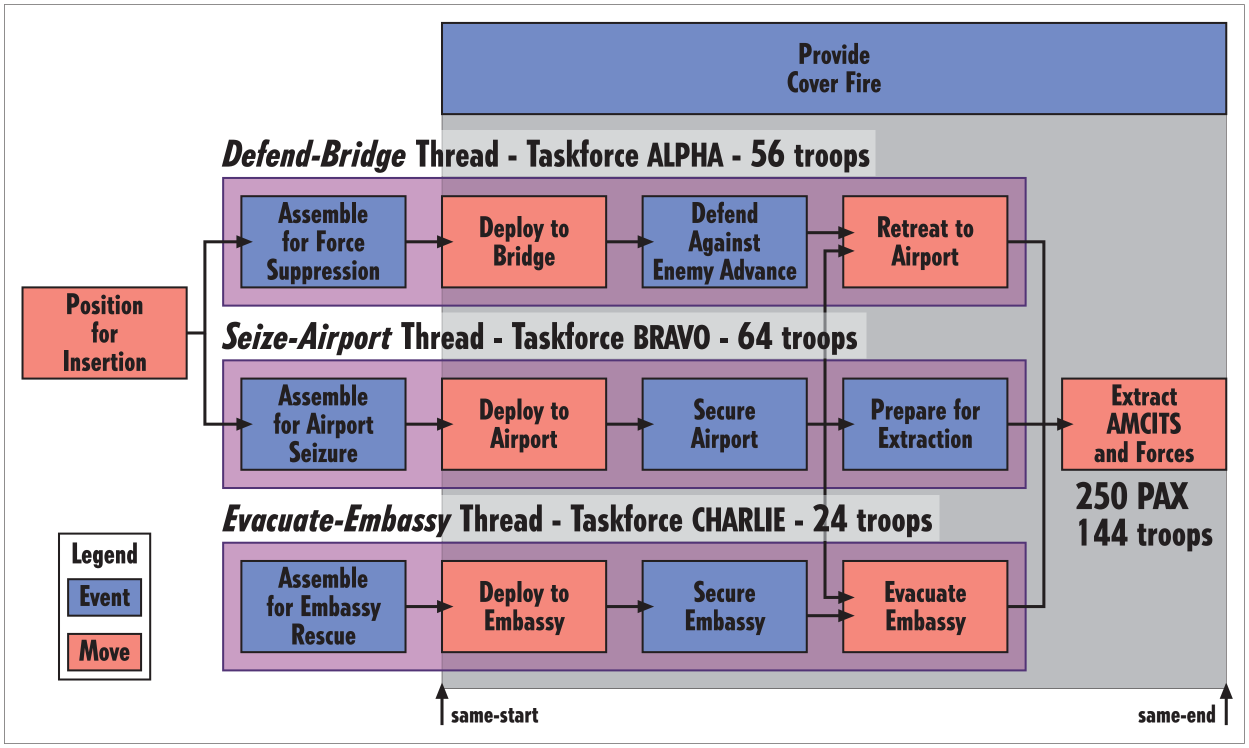

Provide Cover Fire
Defend-Bridge Thread - Taskforce ALPHA - 56 troops
Assemble for Force Suppression
Deploy to Bridge
Defend Against Enemy Advance
Retreat to Airport
Position for Insertion
Seize-Airport Thread - Taskforce BRAVO - 64 troops
Assemble for Airport Seizure
Deploy to Airport
Secure Airport
Prepare for Extraction
Extract AMCITS and Forces
250 PAX
144 troops
Evacuate-Embassy Thread - Taskforce CHARLIE - 24 troops
Legend
Event
Move
Assemble for Embassy Rescue
Deploy to Embassy
Secure Embassy
Evacuate Embassy
same-start
same-end

Figure 29: `Embassy-Rescue` plan

(`Defend-Bridge`) and a final thread aimed at transferring citizens from the embassy (where they have congregated) to the airport (`Evacuate-Embassy`). Once at the airport, everyone is flown out and returned home.

Several of the activities in this plan, specifically the moves, require various transport and combat capabilities. Input constraints also specify initial assumptions concerning the locations and availability of various air and land assets. Additionally, each thread of the plan presumes an independent taskforce, which must be configured from available personnel. Starting from the initial plan, COMIREM assists the user in allocating resources to input activities. In cases where feasible assignment entails the generation of resource support plans (e.g., for positioning an aircraft to the location where it is needed), COMIREM adjusts these activities accordingly as specific resource assignments are considered. If the overall process is successful, the result is a detailed plan where each activity is assigned the resources it requires and is designated to execute in a specified finite time interval.

Using the `Embassy-Rescue` plan as a reference, we now summarize the key functional capabilities provided in COMIREM for intelligently assisting the user in addressing resource-allocation problems of this nature.

3.4.1 Interactive Planning and Resource Allocation

The problem at earlier stages of the planning process is one of reconciling an input concept of operations, represented as a set of inter-related activities, with a heterogeneous set of available resources. Activities specify required capabilities, which may be provided by one or more types of available resources. Figure 30 shows the `Embassy-Rescue` plan as presented by the web-browser-based Comirem GUI. Here we are looking at a "vector" display of the plan, where the background horizontal bars represent locations over time, diagonal lines represent moves from place to place, and horizontal lines represent events occurring at a location.

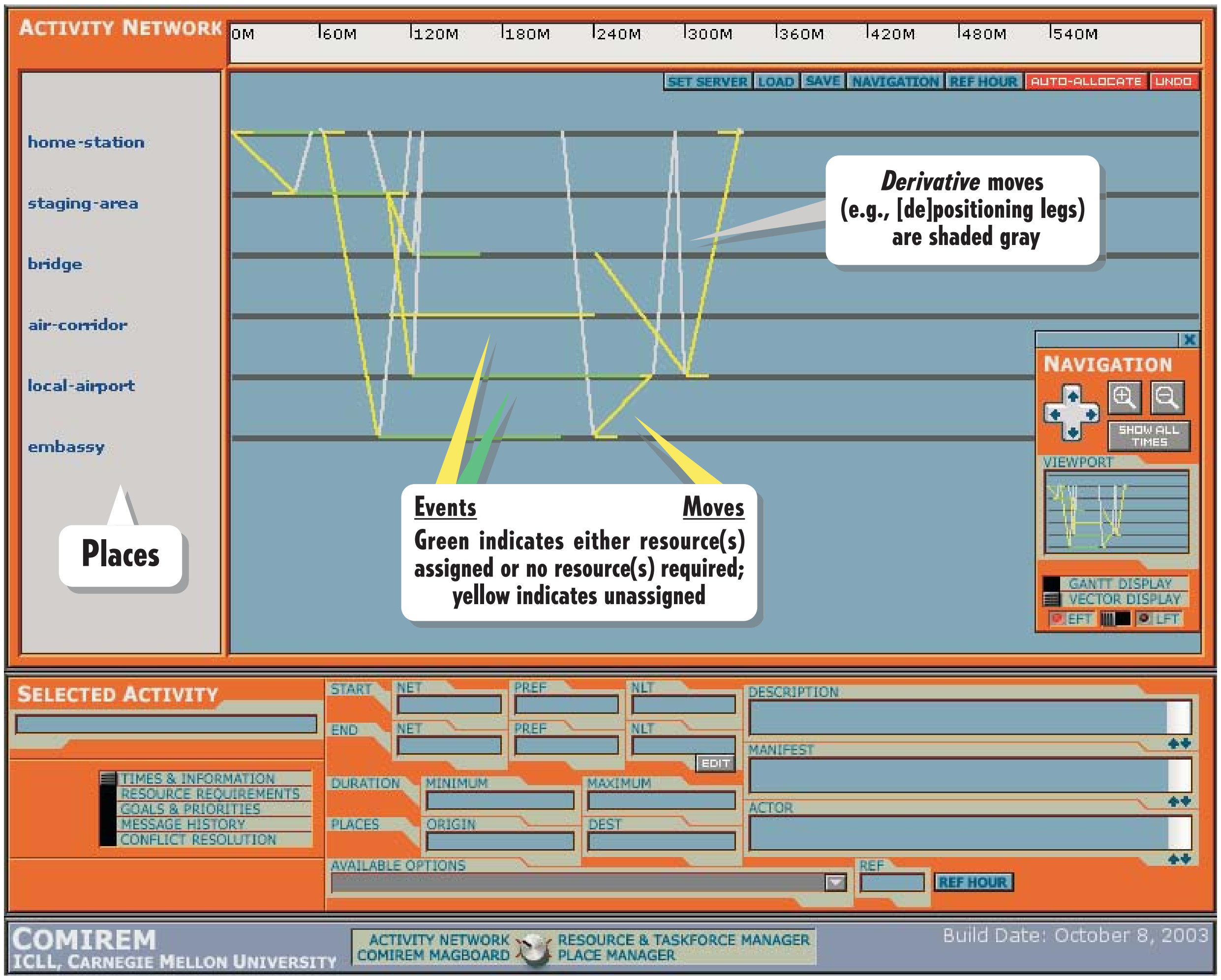
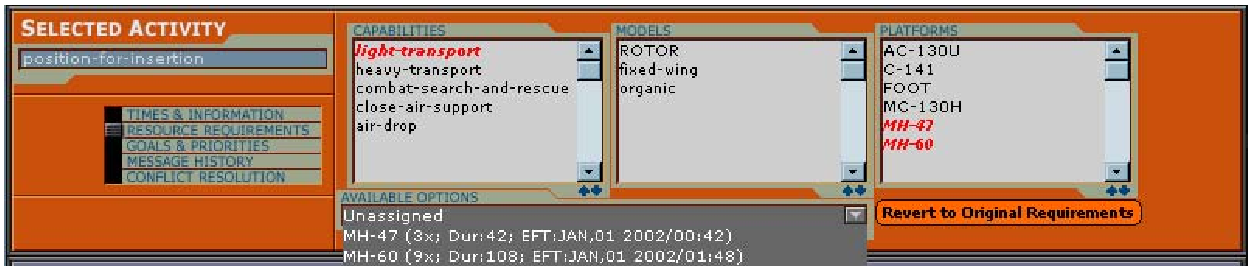


Figure 30: Annotated Comirem GUI display of `Embassy-Rescue` plan

Given this input state, Comirem provides the following support for refining and sourcing a given plan:

- *Option Generation*
  For each unassigned activity in the plan, Comirem maintains the current set of feasible allocation options. Figure 31 shows the situation for the initial `position-for-insertion` activity of the plan. There are two feasible choices: three MH-47 helicopters or nine MH-60 helicopters. The numbers required reflect the size of the manifest to be transported (i.e., the number of troops in the assigned taskforces: ALPHA and BRAVO) and the respective carrying capacities of the assets; the durations given reflect the quantities available, their speed, and their locations.



position-for-insertion

CAPABILITIES
light-transport
heavy-transport
combat-search-and-rescue
close-air-support
air-drop

MODELS
ROTOR
fixed-wing
organic

PLATFORMS
AC-130U
C-141
FOOT
MC-130H
MH-47
MH-60

AVAILABLE OPTIONS
Unassigned
MH-47 (3x; Dur:42; EFT:JAN,01 2002/00:42)
MH-60 (9x; Dur:108; EFT:JAN,01 2002/01:48)

Figure 31: Option generation and selection

- *Visualization of Decision Impact*
  As the user elects to commit to a given option, impact of this assignment is reflected in the plan. Figure 32a, for example, shows the MH-47 assignment (where three helicopters will fly in tandem). Since these assets are initially based at `home-station`, there is no need for a positioning flight; but a depositioning flight has been added. In Figure 32b the impact of alternatively choosing the MH-60's is shown. In this case availability constraints force the activity to be performed in two waves of seven and two trips (and hence the significantly longer duration). A comparison of the utilization of the three MH-47's and the seven MH-60's is shown in Figure 32c.

  Other visual cues provide information about the constrainedness of the current planning search space. The vector display can be re-colored to indicate decision criticality (a function of how many resource assignment options remain). Similarly, the display can be toggled to show the degree of temporal flexibility associated with different activities; Figure 32 actually shows the earliest execution interval of `position-for-insertion`.

- *Requirements and Capabilities Editing*
  Both activity and resource attributes can also be edited to alter the constraints and requirements of the problem. In Figure 31, the `position-for-insertion` activity is shown to require a `light-transport` capability (shown in red), which maps to either of the helicopter types mentioned earlier. Through selection in this display, the user is free to adjust these requirements, either specifying alternative or additional capabilities or platforms that might be viable to consider. Any adjustments result in a corresponding update of feasible options. Other timing constraints on activities (e.g., deadlines or synchronization points) can similarly be manipulated to open up (or restrict) the set of options.

  Resource attributes can similarly be manipulated to vary basic problem assumptions. For example, the numbers of resources available and their initial locations can be revised to evaluate the consequences of an alternative deployment. In Figure 33 we show an editing action to introduce the constraint that only one aircraft can be on the ground at a time at location `staging-area` (whereas it had previously been specified as *unconstrained*). Figure 34 shows the consequence on the `position-for-insertion` activity: the three MH-47 flights now must unload in a staggered fashion.

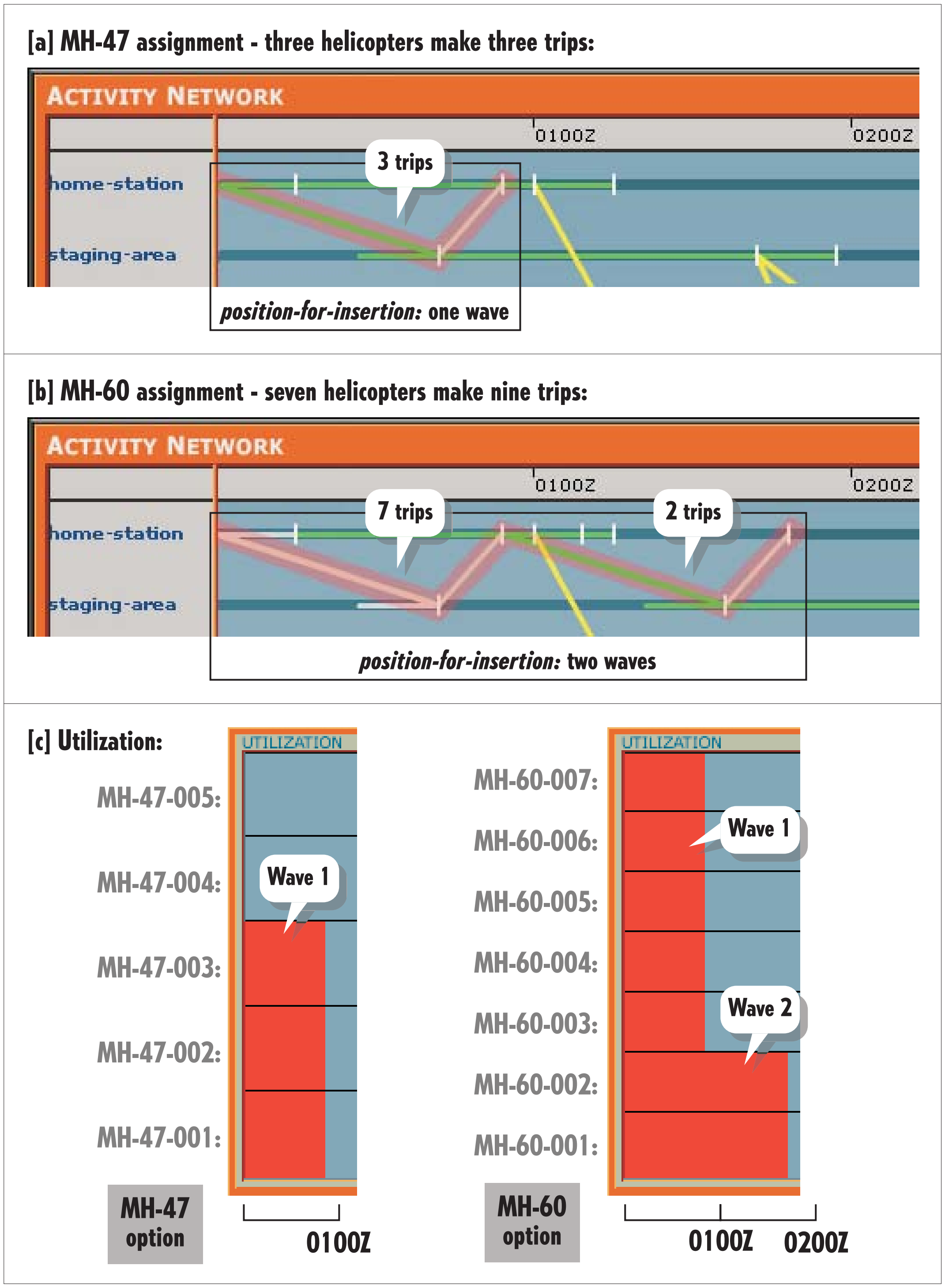
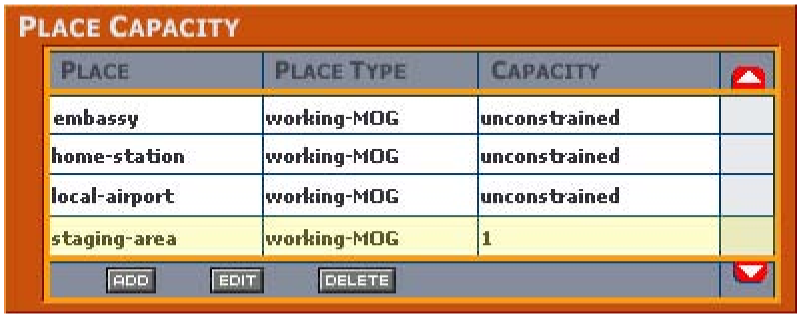


Figure 32: MH-47 and MH-60 assignment options and utilization profiles for the `position-for-insertion` activity


PLACE CAPACITY

PLACE	PLACE TYPE	CAPACITY
embassy	working-MOG	unconstrained
home-station	working-MOG	unconstrained
local-airport	working-MOG	unconstrained
staging-area	working-MOG	1

Figure 33: Imposing a maximum-on-ground (MOG) constraint

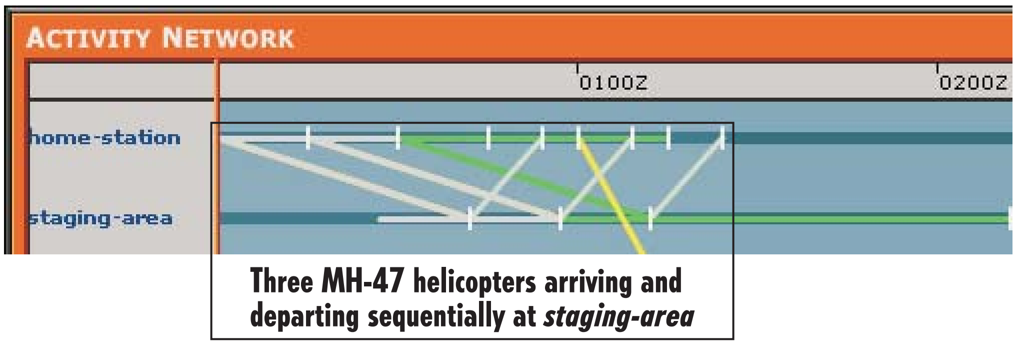

ACTIVITY NETWORK
0100Z
0200Z
home-station
staging-area
Three MH-47 helicopters arriving and departing sequentially at *staging-area*

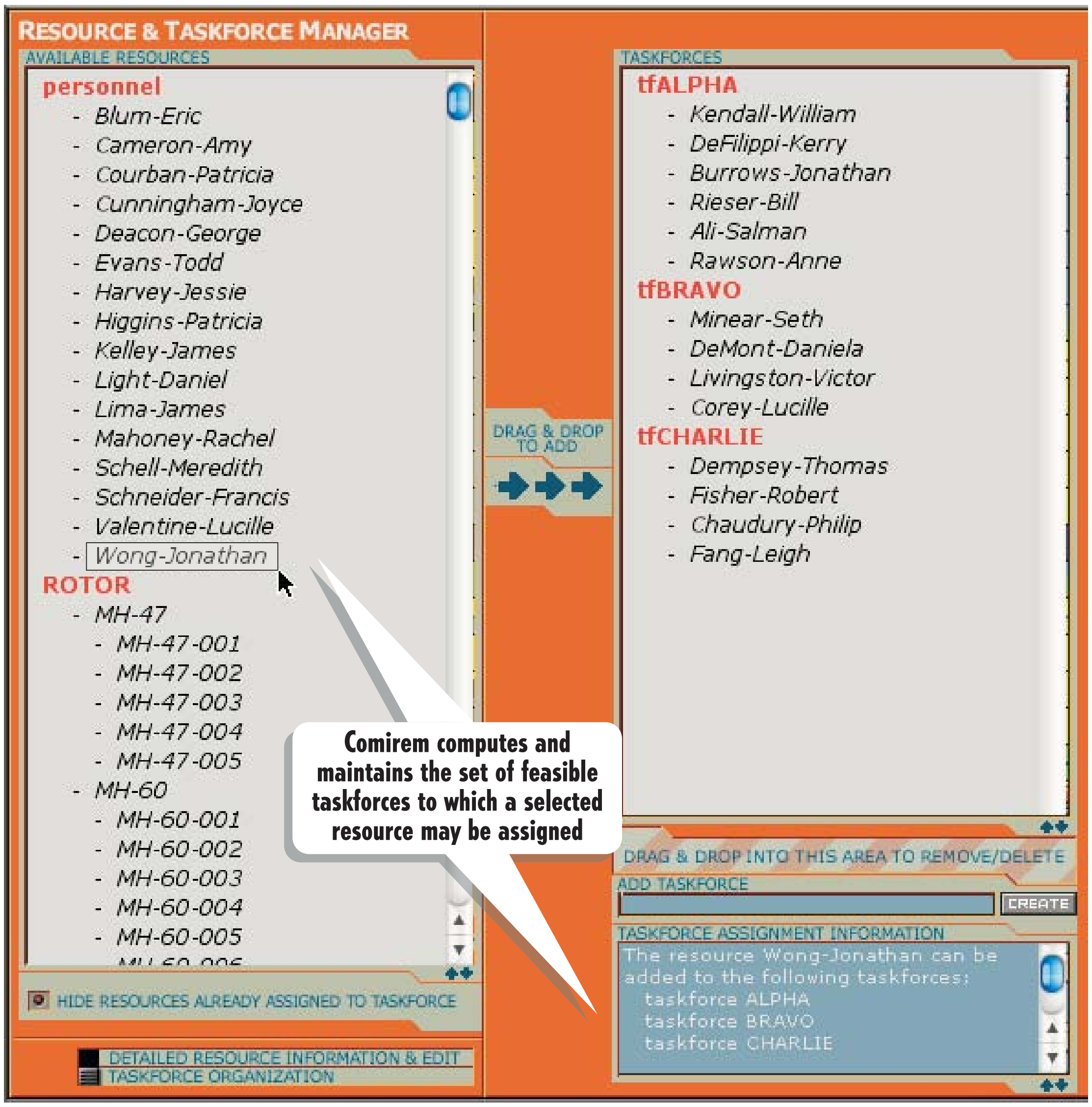
Figure 34: Revised MH-47 assignment for the `position-for-insertion` activity following the tightening of the `staging-area` MOG constraint

- *Automated Assignment and Feasibility*
  The user can selectively rely more on system decision procedures. For any selected set of unassigned activities in the plan, the user can invoke an underlying automatic scheduler to generate a feasible assignment. This action constitutes a check of the overall resource feasibility of the plan (assuming whatever decisions have been previously fixed by the user). The allocation goals and preferences that focus the automated scheduler can be adjusted by the user to produce different sets of resource assignments, in scenarios where overall resource capabilities contain this flexibility.

- *What-If Analysis*
  As implied by much of the discussion above, most constraints and planning decisions are pliable through the interface to allow general exploration of the problem and solution space. The system provides a general ability to undo any user-level action (or sequence of actions) previously taken. Coupled with the above described capabilities, this mechanism provides a flexible, open-ended framework for what-if analysis.

3.4.2 Resource Configuration

Complementary capabilities for interactively configuring aggregate resources (taskforces in our `Embassy-Rescue` scenario) from more primitive resource objects (in this case, individual personnel) are provided through a drag and drop interface as shown in Figure 35. When a resource is added to an aggregate under development, the system checks basic availability constraints. When an aggregate resource is assigned to a thread, this resource and all of its constituents become otherwise unavailable for allocation over the period that the thread executes. Hence, just as in the case where primitive resources are allocated, the potential addition of a resource to an aggregate resource or the potential assignment of an aggregate resource to a thread may be infeasible due to prior commitments, or likewise, may require additional sequencing of competing activities in the plan.


RESOURCE & TASKFORCE MANAGER
AVAILABLE RESOURCES
TASKFORCES
DRAG & DROP TO ADD
Comirem computes and maintains the set of feasible taskforces to which a selected resource may be assigned
DRAG & DROP INTO THIS AREA TO REMOVE/DELETE
ADD TASKFORCE
CREATE
TASKFORCE ASSIGNMENT INFORMATION
HIDE RESOURCES ALREADY ASSIGNED TO TASKFORCE
DETAILED RESOURCE INFORMATION & EDIT
TASKFORCE ORGANIZATION

personnel
- Blum-Eric
- Cameron-Amy
- Courban-Patricia
- Cunningham-Joyce
- Deacon-George
- Evans-Todd
- Harvey-Jessie
- Higgins-Patricia
- Kelley-James
- Light-Daniel
- Lima-James
- Mahoney-Rachel
- Schell-Meredith
- Schneider-Francis
- Valentine-Lucille
- Wong-Jonathan

ROTOR
- MH-47
  - MH-47-001
  - MH-47-002
  - MH-47-003
  - MH-47-004
  - MH-47-005
- MH-60
  - MH-60-001
  - MH-60-002
  - MH-60-003
  - MH-60-004
  - MH-60-005

tfALPHA
- Kendall-William
- DeFilippi-Kerry
- Burrows-Jonathan
- Rieser-Bill
- Ali-Salman
- Rawson-Anne

tfBRAVO
- Minear-Seth
- DeMont-Daniela
- Livingston-Victor
- Corey-Lucille

tfCHARLIE
- Dempsey-Thomas
- Fisher-Robert
- Chaudury-Philip
- Fang-Leigh

```
The resource Wong-Jonathan can be
added to the following taskforces:
  taskforce ALPHA
  taskforce BRAVO
  taskforce CHARLIE
```

Figure 35: Annotated taskforce composition form

3.4.3 Conflict Analysis and User-Oriented Explanation

User acceptance of system decisions in successful problem-solving episodes and effective user involvement in circumstances where a system reaches a problem-solving impasse both require that a system be comprehensible. This requirement, in turn, implies that a system be capable of explaining its decisions in user-understandable terms. Our previous work has argued for the use of a scheduling domain ontology as a basis for translating user problem specifications into internal system models [Smith and Becker, 1997]. Recent work with COMIREM has focused on a complementary use of a domain ontology to compute user-oriented explanations of system decisions and conflicts [Smith *et al.*, 2004]. The central idea is to use COMIREM domain ontology knowledge to (1) compute the set of user-relevant and manipulable constraints that form an explanation, (2) identify various constraint-relaxation options that the user might consider to resolve a conflict, and (3) provide content for generating user-understandable explanations of conflicts and possible resolving actions.

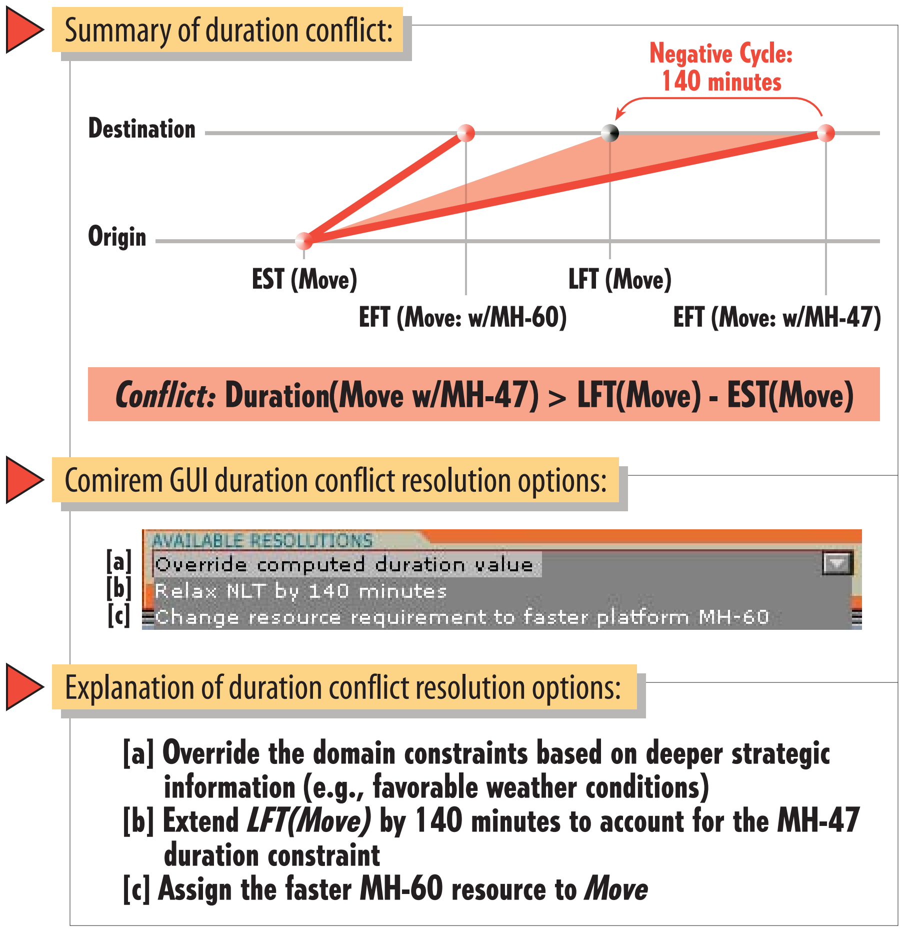
Figure 36 shows the system's response in a situation where it has detected that it is not possible to complete an activity by its imposed deadline if the assigned resource is used. In this case, knowledge relating temporal constraints in the underlying STP constraint network to various domain level constructs (e.g., activity and relation types) is used to filter the set of constraints found to be in conflict by the temporal constraint propagator and identify the subset that is user manipulable. Once this subset of constraints is identified, knowledge associated with corresponding domain entities is used to propose possible constraint relaxation options. The user can then choose from this set of proposed options.

3.4.4 Execution Management

Finally, COMIREM provides additional tools for managing resources and resource assignments as execution unfolds:

- *Resource Tracking*
  A "magboard" UI display (shown in Figure 37) provides a geographical visualization of the location of all resources at any given point in time. The display distinguishes those resources that are (1) in-transit from one location to another, (2) at a location but engaged in some activity (e.g., providing air cover support, securing an airport), or (3) at a location and available for use.

- *Plan Tracking*
  Both the vector display of the plan (and an additional Gantt-based view) can also be used to visualize the current state of execution. As information about the actual start and end of activities is received, the executing frontier of the plan is highlighted and any conflicts between the actual and expected course of events are signaled.

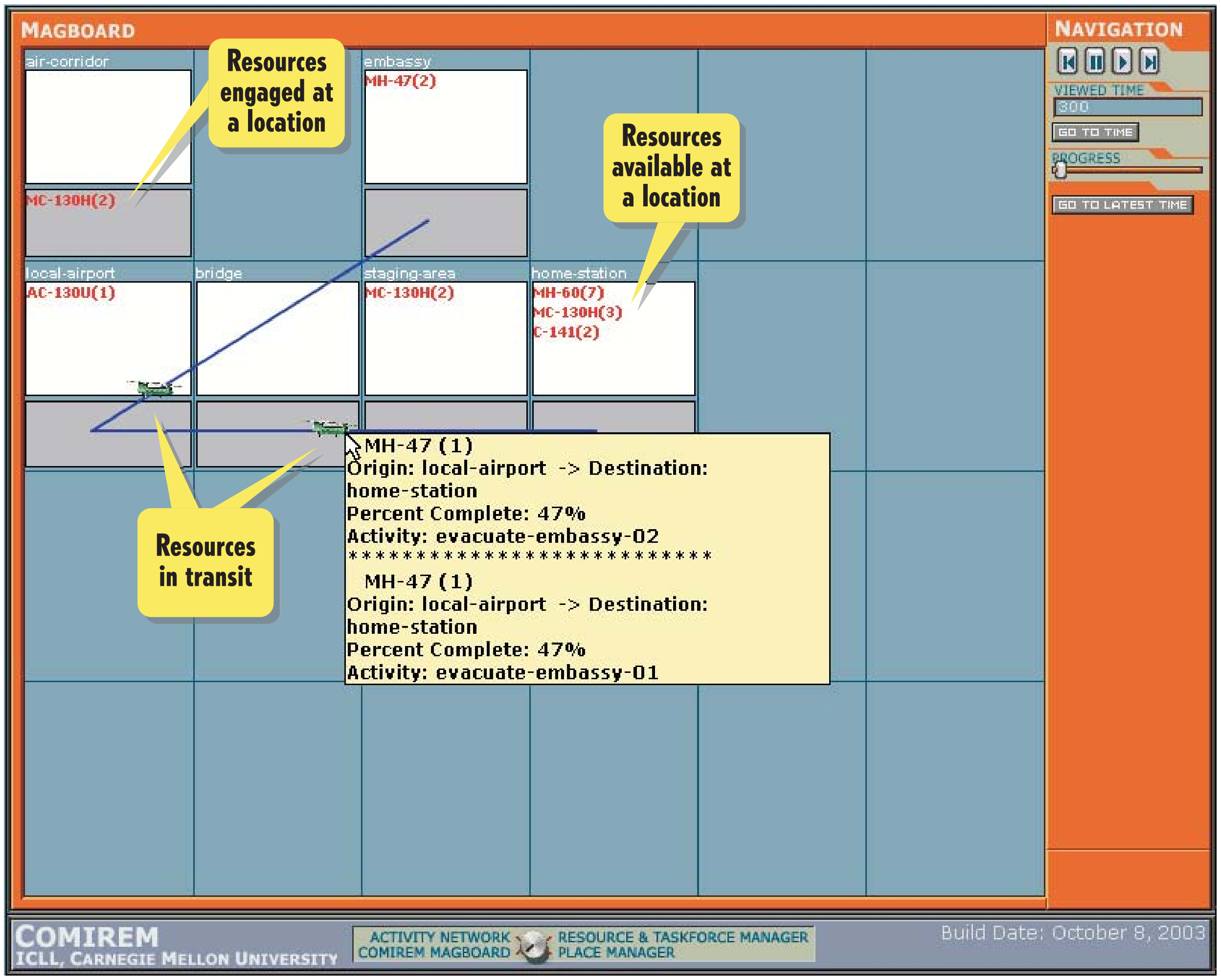

Summary of duration conflict:
Negative Cycle: 140 minutes
Destination
Origin
EST (Move)
EFT (Move: w/MH-60)
LFT (Move)
EFT (Move: w/MH-47)
Conflict: Duration(Move w/MH-47) > LFT(Move) - EST(Move)
Comirem GUI duration conflict resolution options:
AVAILABLE RESOLUTIONS

[a] Override computed duration value
[b] Relax NLT by 140 minutes
[c] Change resource requirement to faster platform MH-60

Explanation of duration conflict resolution options:

[a] Override the domain constraints based on deeper strategic information (e.g., favorable weather conditions)
[b] Extend *LFT(Move)* by 140 minutes to account for the MH-47 duration constraint
[c] Assign the faster MH-60 resource to *Move*

Figure 36: A sample duration conflict and corresponding COMIREM-generated resolution options



```
  MH-47 (1)
Origin: local-airport  -> Destination:
home-station
Percent Complete: 47%
Activity: evacuate-embassy-02
**************************
  MH-47 (1)
Origin: local-airport  -> Destination:
home-station
Percent Complete: 47%
Activity: evacuate-embassy-01
```

Figure 37: Annotated Comirem GUI Magboard screen shot

3.5 Scheduler Architecture

An overview of the underlying COMIREM system architecture is presented in Figure 38. The planning/scheduling engine is implemented in Common Lisp [Steele Jr., 1990] as an extension of the platform-independent Ozone scheduling framework [Smith *et al.*, 1996].[20] It operates together (i.e., as a sibling Lisp process) with a Common Lisp HTTP server,[21] which facilitates the transmission of either: (1) dynamically generated HTML/XML responses or (2) files from disk (e.g., HTML/XML documents) to an Internet browser client using standard CGI (Common Gateway Interface), and is used to support both visualization and interactive development of plans and schedules. The coupling of the core planner/scheduler with a web server provides immediate interoperability with most any web browser (and by extension, nearly any hardware platform), and even more importantly, the broad range of existing interactive visualization tools and applications designed to work with them. The COMIREM system is therefore able to take advantage of the wealth of web-based visualization tools that already exist and allow the user to interact with it using whatever tool is best suited for the present circumstances (e.g., hardware considerations, user preference).

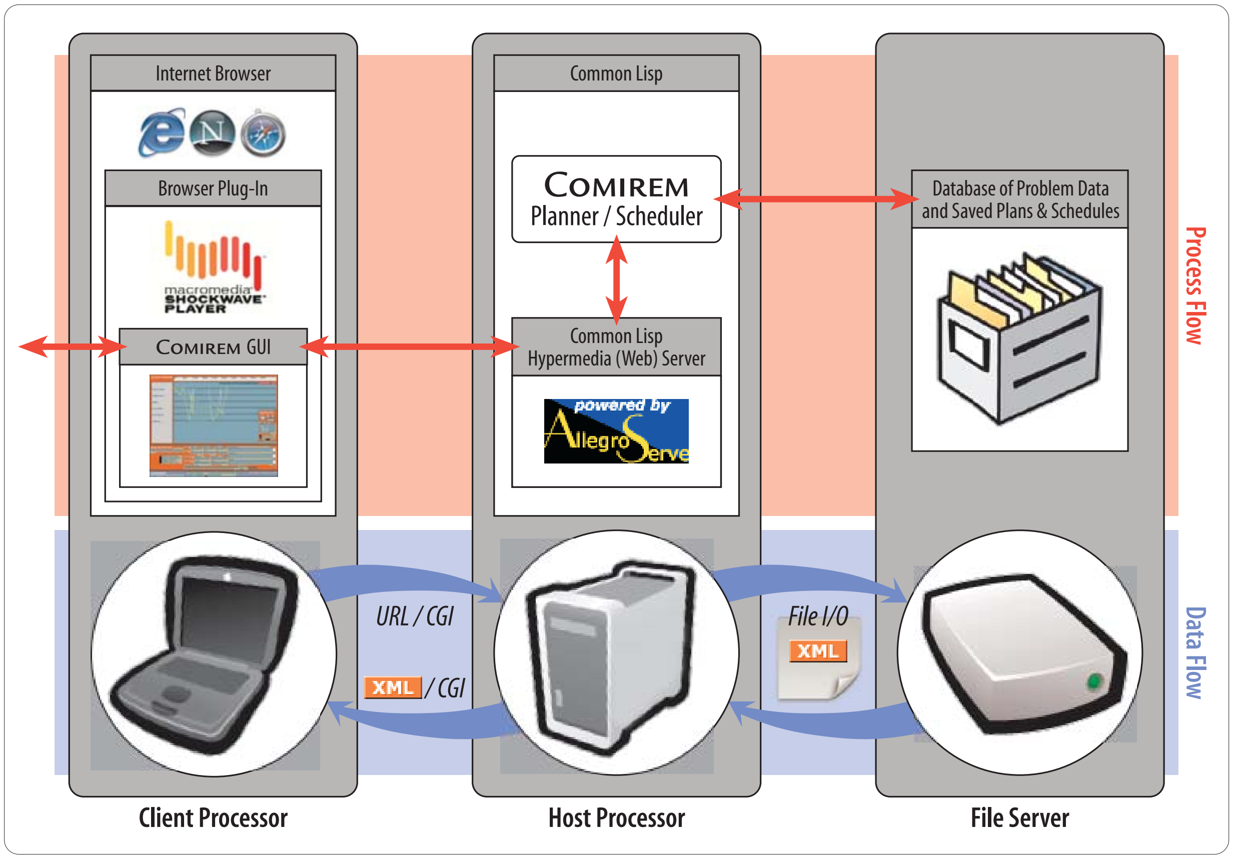


Figure 38: COMIREM system architecture showing process and data flows

[20] COMIREM runs on the Apple MacOS X, Microsoft Windows and Sun Solaris platforms.
[21] Both AllegroServe in Allegro Common Lisp and CL-HTTP in Macintosh Common Lisp are supported.

COMIREM is presently capable of interacting with three different types of visualization and interaction tools.

- A Shockwave application serves as the primary user interface component (see the *Client Processor* on the left in Figure 38). This choice reflects an underlying objective to explore the potential of more modern web-based design tools for interactive visualization and manipulation of plans and schedules. This application was built using Macromedia's Director program and the Lingo scripting language, and is executable in any common web browser using the freely available Shockwave plug-in application.

- The architecture also supports the use of standard HTML/XML forms as an information display and input medium. We envision such interface components as being particularly necessary in some mobile circumstances, where screen real estate and computational considerations may prohibit extensive graphical interface capabilities. Within the current COMIREM interface, forms are provided for adding, retracting and editing various plan constraints and resource assignments.

- Finally, provision is made to interact and interoperate with externally developed Java applets.[22] In general, our current user interface design perspective is to de-emphasize Java components in favor of an expanded functionality Shockwave application. While interoperability with Java applets is important given their widespread use in web application environments, our experience to date nonetheless indicates that Shockwave offers the potential for much more lightweight software components (i.e., smaller footprint, no run-time library support required).

Each of these interfaces communicates with the COMIREM server via standard HTTP protocol (currently HTTP/1.1). Requests from browser clients are encoded as URLs and transmitted to the server, and HTML/XML responses are then generated (or uploaded from disk) and returned to the client web browser using CGI. (Direct peer-to-peer socket communication with the COMIREM server is also supported, in case interaction with an encapsulated Java-based applet is required.) For the most part, interaction with the COMIREM planner/scheduler is simply achieved by launching a web browser and requesting a specific web page.

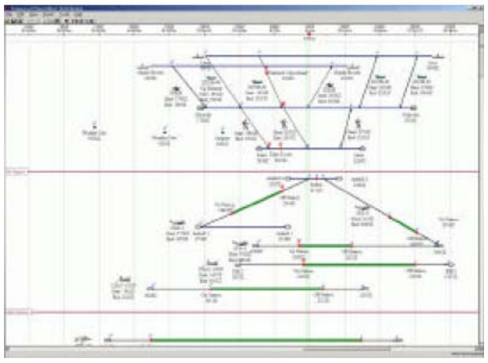
[22] An early version of COMIREM relied on the use of Java applets adapted from another Ozone-based system to display resource capacities.

4 Interoperability with the SOFTools Environment

Comirem has been integrated with a collection of other tools developed for the SOF domain, including SOFTools (an interactive plan-authoring tool) and C2PC (an interactive mapping program). Comirem accepts XML-formatted plan data files from SOFTools, performs resource-feasibility-checking on the plan, and exports "magboard"-style output data for display in C2PC.

4.1 Temporal Plan Editor

SOFTools is a collection of mission planning and editing tools developed by BBN (initially) and General Dynamics Advanced Information Systems to allow a (human) planner to create and analyze plans in the SOF domain. The planner establishes the temporal representations of a mission plan and any related supporting plans using SOFTools' *Temporal Plan Editor* by defining movements and events at various locations and times. These plans represent a high-level view of a developing plan, in which the times are essentially estimates based on expected plan locations and resource assignments. Interaction with Comirem begins with the translation and import of the SOFTools-generated plan file, which permits Comirem to evaluate both the resource and temporal feasibility of the SOFTools plan under a range of resource-availability situations.


Temporal Plan Editor screen shot

The key features of the interaction between SOFTools involve the translation of the input data and the resource and temporal feasibility-checking, summarized below:

- *Translation of input data:*

  SOFTools allows the planner to assign estimated departure and arrival times to movements and events in a plan and to attach certain resource quantities to these activities—without any consideration of actual resource availability. There are no explicit temporal sequencing constraints and the anticipated activity durations are all estimates that are based on expected locations and typical resource traveling speeds. For the purpose of analyzing this data, Comirem has been equipped with a module that translates—with some necessary modification—SOFTools plan data into Comirem-acceptable form.

- *Resource and temporal feasibility-checking:*

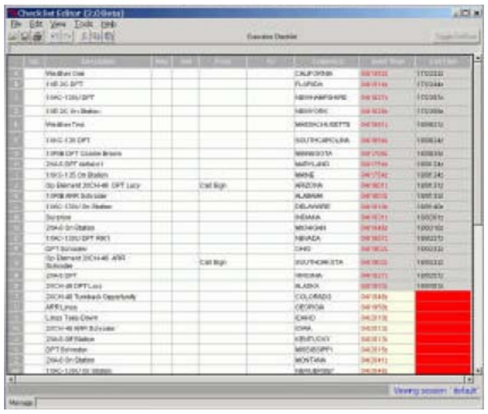
  Once a SOFTools plan has been translated into Comirem format (a process that is performed automatically whenever a SOFTools plan (scenario) is loaded), Comirem is ideally suited to perform the necessary resource and temporal feasibility checks to assure that a plan is generally feasible given a particular resource-availability situation. The following checks are performed:

  – *Resource availability* is evaluated by: (1) ensuring that all required resources are, or will be, placed at the locations required by the activities to which they have been assigned, and (2) that sufficient resource capacity is available to satisfy the desired assignments.

- *Temporal feasibility* is evaluated by applying a more strict domain model to the estimated activity durations that relies on detailed resource models and more accurate distance-based movement-duration calculations.

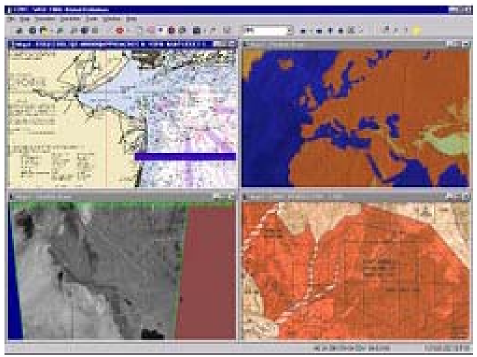
4.2 Check List Editor

Interaction with SOFTools' *Check List Editor* (CLE) provides an additional means for assessing feasibility in plans, by adding the possibility of having to consider actual execution results. The CLE allows planners to augment a SOFTools plan with runtime information that indicates (among other things) when an activity has begun or completed execution. Comirem is capable of accepting *calltimes* (i.e., actual start and end times for activities) and factoring those times into its resource and temporal feasibility-checking routines. Calltimes are treated by Comirem as hard (i.e., non-relaxable) temporal constraints that compress activity time windows down to the absolute minimum. Additionally, the processing of calltimes causes Comirem to update its internal clock, which prohibits it from reconsidering decisions made in the past, and further tightens the temporal flexibility of the plan.


Check List Editor screen shot

4.3 Command and Control Personal Computer

Command & Control Personal Computer (C2PC), developed by Northrop Grumman, is an interactive mapping application that provides a geographically based situational-awareness capability that can be used to track assets over time in support of operational and tactical decision-making. Comirem interacts with C2PC using *overlays*, which are data files that are continually monitored by C2PC and (re)displayed immediately when modified. Overlay files allow the changing positions of assets to be displayed in animated form over time. Using its own Magboard display (see Figure 37, Section 3.4.4 (**Execution Management**), page 57, and Section B.4.4), Comirem can be made to generate a sequence of overlay files at specified time increments so that C2PC can display the plan as its execution is simulated.


C2PC screen shot

5 Demonstration Problems

As part of the evaluation process for COMIREM, two different SOF-type scheduling problem scenarios have been developed to help assess its modeling and scheduling capabilities. The first, **Embassy Rescue** (developed within the Intelligent Coordination and Logistics Laboratory at CMU and introduced earlier in Section 3.4, **Mixed-Initiative Resource Management**), involves a rescue mission to a U.S. embassy in a foreign country using mainly air assets. The second, **Exotic Dancer** (provided by the Active Templates program via the SOFTools Temporal Plan Editor), involves a synchronized attack on two remote targets using a combination of air, sea and land assets. Both of these scenarios are described in this section. Section 5.1 presents some common type definitions shared by both problems. Section 5.2 describes the **Embassy Rescue** scenario, and Section 5.3 describes the **Exotic Dancer** scenario.

One major difference between the two scenarios is the data-loading process. For **Embassy Rescue**, the data files are loaded directly without any additional translation. For **Exotic Dancer**, however, since the scenario data comes directly from XML output files generated by SOFTools' Temporal Plan Editor, an additional translation phase must be performed to first generate a *scenario*-`plan.xml` file that is amenable to COMIREM. For both scenarios, corresponding situation data files defining the necessary resource and activity types, capabilities (if necessary), and resource instances have been established to facilitate system evaluation.

Additional differences between the two scenarios involve: (1) the use of constraints, including temporal sequencing constraints and earliest and latest start and finish constraints on activities, (2) the use of capabilities as a means of connecting required resources to activities, and (3) the use of both capacitated and uncapacitated place component resources. These differences are described in greater detail as part of the discussion of each scenario.

5.1 Common Definitions

Both the **Embassy Rescue** and **Exotic Dancer** scenarios rely heavily on the use of air assets for transporting personnel. A shared activity type supports the sourcing of aircraft, and the shared resource types include an `:organic FOOT` resource and two common helicopter (i.e., `ROTOR`) types.

5.1.1 Activity Types

The `RESOURCE-SOURCING` wrapper activity type presented in Figure 39 and discussed extensively in Section 3.2.1.1 (**Hierarchical Activity Networks**), facilitates the sourcing of available mobile resources (i.e., aircraft *and* seacraft) from elsewhere in the environment. The wrapper surrounds a nexus activity (the `:child` activity referenced in Figure 39)—typically a `move` activity of some sort—with a preceding positioning leg (a `move` activity) and a succeeding depositioning leg (another `move` activity) that can be used for the sourcing and return of the resource. The instantiated `RESOURCE-SOURCING` wrapper will permit COMIREM to look for candidate resources according to how the actual nexus activity is defined—i.e., based on the values of its `resource-origin` and `resource-destination` variables (specified in its `initargs` tag). If these variables are set to specific places or locations, then only resources positioned at, and able to return to those locations, are feasible. If these variables are left unbound (i.e., free), then resources from anywhere in the environment may be used.

5.1.2 Resource Types

Figure 40 presents the XML specifications for three resource types shared by the **Embassy Rescue** and **Exotic Dancer** scenarios. The two helicopter types, `MH-47` and `MH-60`, are common SOF air assets. The `FOOT` resource is universal, representing movement by *foot* (and making use of the `:organic` model type). In both scenarios, a `UNIVERSAL-FOOT` resource is instantiated from this type, providing unlimited capacity.

5.2 Embassy Rescue

As described in Section 3.4 (**Mixed-Initiative Resource Management**) and illustrated in Figures 28 (page 48) and 29 (page 49), the **Embassy Rescue** scenario describes a rescue mission that culminates in the evacuation of personnel from a U.S. embassy on foreign soil, and consisting of three initially separate threads that converge at the end. The three threads behave as follows:

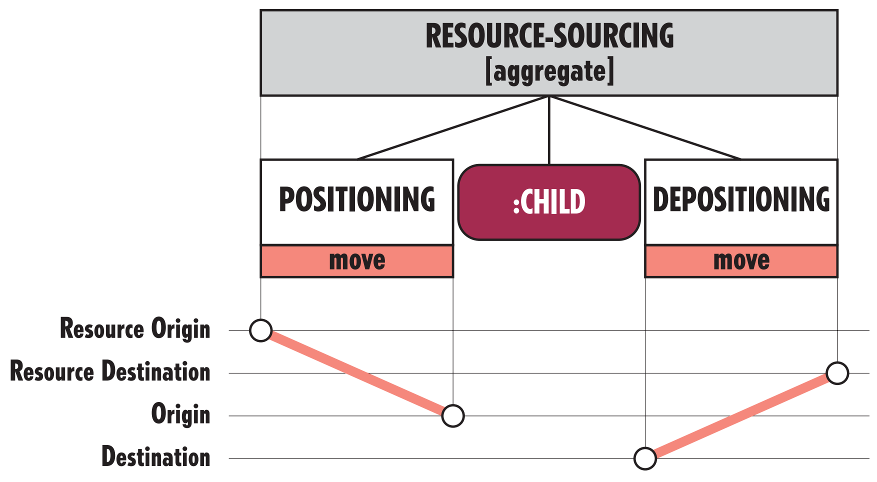
1. *Defend-Bridge* - A taskforce (ALPHA), deploys to a strategic bridge location within enemy territory to defend a temporarily enemy-controlled airport against further enemy advance, before moving to the newly secured airport for final evacuation.

2. *Seize-Airport* - A taskforce (BRAVO), deploys to the enemy-controlled airport, overtakes enemy forces there, and then prepares for the evacuation of the rescued embassy personnel and all three taskforces.

3. *Evacuate-Embassy* - A taskforce (CHARLIE), deploys to the embassy, secures it, and then evacuates all personnel to the recently secured airport for final evacuation.

Once all of the personnel and taskforces have assembled at the airport, a single movement (in multiple waves) evacuates everyone back to the home station. And throughout the main body of the plan, a single `provide-cover-fire` activity does precisely that.

The internal representation of this plan is presented in Figure 41, showing all of the activities and locations, derivative sourcing legs, temporal sequencing constraints, cargo manifests, required durations and resource requirements. Places in the plan (some of which are repeated for the sake of clarity) are identified along the y-axis, extending horizontally across the figure. The 10-hour plan horizon extends across the x-axis. Activities are drawn in green (for moves) and yellow (for events), with derivatives drawn in dark blue. The activities are labeled with boxes showing their required capability or resource class at the top, the name of the activity in the middle, and the manifest, taskforce identification, and sometimes the required duration at the bottom. The temporal sequencing constraints are drawn with red annotated arrows.



```
<type>
  <wrapper name="resource-sourcing">
    <variables>resource-origin resource-destination</variables>
    <inputs>origin destination</inputs>
    <decomposition>
      <child-spec name="positioning">
        <type-ref>move</type-ref>
        <initargs>
          <initarg name="origin">
            <slot>resource-origin</slot>
          </initarg>
          <initarg name="destination">
            <keyword>origin</keyword>
          </initarg>
        </initargs>
      </child-spec>
      <child-spec name="depositioning">
        <type-ref>move</type-ref>
        <initargs>
          <initarg name="origin">
            <keyword>destination</keyword>
          </initarg>
          <initarg name="destination">
            <keyword>resource-destination</keyword>
          </initarg>
        </initargs>
      </child-spec>
    </decomposition>
    <constraints>
      <constraint>
        <type-ref>successor</type-ref>
        <from>positioning</from>
        <to>:child</to>
        <ub><interval><minutes>0</minutes></interval></ub>
      </constraint>
      <constraint>
        <type-ref>successor</type-ref>
        <from>:child</from>
        <to>depositioning</to>
        <ub><interval><minutes>0</minutes></interval></ub>
      </constraint>
    </constraints>
  </wrapper>
</type>
```

Figure 39: Structure and XML specification of the RESOURCE-SOURCING wrapper activity type

MH-47

MH-60

```
<type>
  <aircraft name="MH-47">
    <description>
      Chinook: infiltration, exfil-
      tration, air assault, resupply
      and sling operations
    </description>
    <model-type>ROTOR</model-type>
    <speed units="mph">177</speed>
    <configurations>
      <configuration name=":default">
        <cargo max="55">PAX</cargo>
      </configuration>
    </configurations>
  </aircraft>
</type>
```

```
<type>
  <aircraft name="MH-60">
    <description>
      Black Hawk: infiltration, exfil-
      tration, resupply of special op-
      erations forces, and CSAR
    </description>
    <model-type>ROTOR</model-type>
    <speed units="mph">184</speed>
    <configurations>
      <configuration name=":default">
        <cargo max="14">PAX</cargo>
      </configuration>
      <configuration
        name="close-air-support">
        <cargo max="1">MUNITIONS</cargo>
      </configuration>
    </configurations>
  </aircraft>
</type>

<type>
  <landcraft name="FOOT">
    <model-type>:organic</model-type>
    <speed units="mph">4</speed>
  </landcraft>
</type>
```

Figure 40: XML specifications of shared resource types

The **Embassy Rescue** scenario is notable for a number of reasons, alluded to earlier:

1. *Temporal sequencing constraints:*

   All of the activities in the plan are free to shift temporally within a 10-hour scheduling horizon. The necessary synchronization of activities is achieved using temporal sequencing constraints, such as `successor`, `same-start` and `same-finish`, without any ref-hours or anchors. These constraints also ensure that the `provide-cover-fire` activity properly encompasses its dependent plan activities.

2. *Capabilities:*

   A number of the resource requirements for activities in the plan rely on user-defined domain-specific capabilities (e.g., `light-transport`, `air-drop`) instead of specific resource classes.

3. *Capacitated and uncapacitated place component resources:*

   Capacity for aircraft at the embassy is provided through two capacitated `working-MOG` components (`embassy-yard-1` and `embassy-yard-2`), one of which must be secured by each aircraft doing any loading or unloading of cargo on the site. Capacity for aircraft loading and unloading elsewhere in the plan is represented using *uncapacitated* (i.e., infinite capacity) components, such as at the enemy airport and home station.

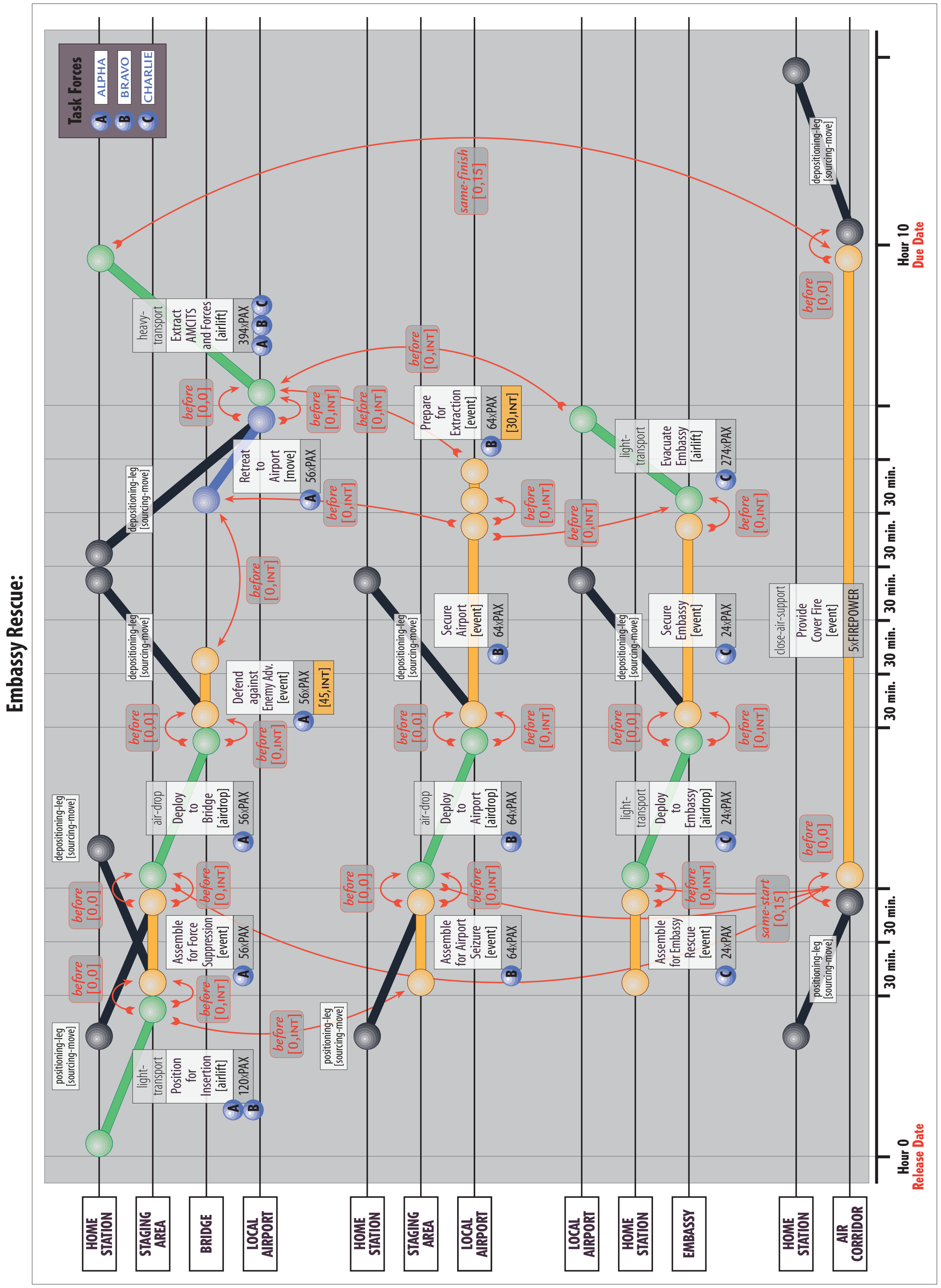


Figure 41: Embassy-Rescue activity network

5.2.1 Activity Types

The **Embassy Rescue** domain utilizes five activity types: the `load-step` and `unload-step` events for handling cargo, the `airlift` and `airdrop` aggregate activities for moving cargo (in this case, by air), and the `event-w-sourcing` wrapper activity that is used to represent a patrol activity such as the aforementioned `provide-cover-fire`. The following subsections describe these types in further detail.

5.2.1.1 Cargo Load and Unload

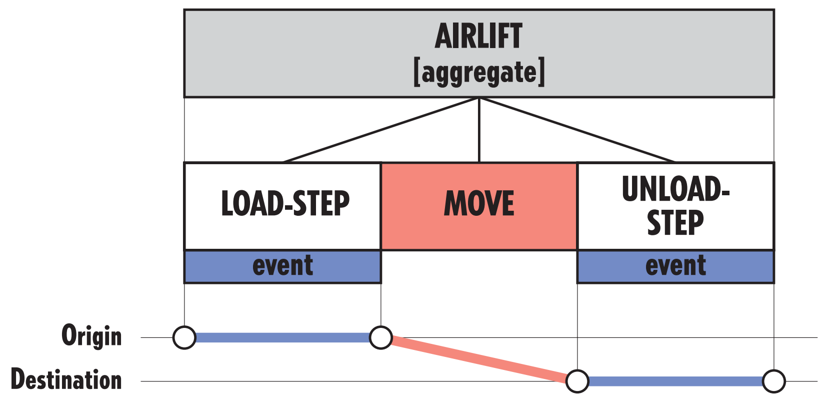
Cargo loading and unloading is handled by means of the `load-step` and `unload-step` event types, respectively. The XML specifications for both are shown in Figure 42. Each basically requires a sufficient capacity of a `working-MOG` resource class instance, which refers to an area of some kind that can accommodate the loading or unloading of cargo from a resource. The `AIRCRAFT` cargo type is used (currently) to indicate the maximum capacity of a `working-MOG` resource instance (which, in **Embassy Rescue**, is 1).

```
<type>
  <event name="load-step">
    <requirements>
      <requirement>
        <resource-class>working-MOG</resource-class>
      </requirement>
    </requirements>
  </event>
</type>

<type>
  <event name="unload-step">
    <requirements>
      <requirement>
        <resource-class>working-MOG</resource-class>
      </requirement>
    </requirements>
  </event>
</type>
```

Figure 42: XML specification of `Embassy-Rescue` cargo-[un]loading activity types

5.2.1.2 Airlift

The `airlift` activity type defines an activity network structure similar to that of the `RESOURCE-SOURCING` wrapper described in Section 5.1.1 (**Activity Types**), except that instead of wrapping positioning legs around a nexus activity, the `airlift` activity expands into three subactivities: a `load-step`, a basic `move`, and an `unload-step`. The `working-MOG` resource classes required by the (un)load steps must come from the `origin` location for `load-step`, and from the `destination` location for `unload-step`. A summary of the `airlift` activity type is provided in Figure 43.

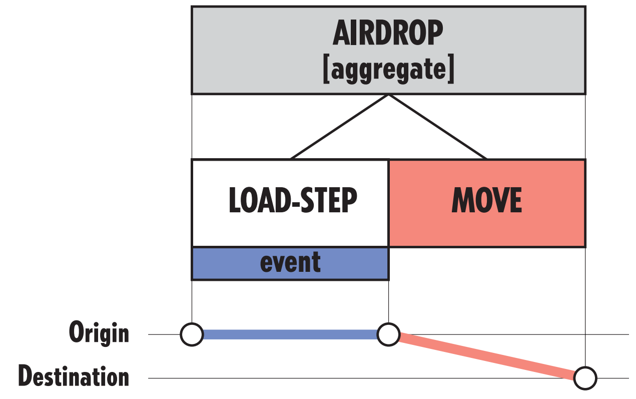


```
<type>
  <move name="airlift">
    <decomposition>
      <child-spec>
        <type-ref>load-step</type-ref>
        <initargs>
          <initarg name="place"><slot>origin</slot></initarg>
        </initargs>
      </child-spec>
      <child-spec>
        <type-ref>move</type-ref>
        <initargs>
          <initarg name="origin">
            <slot>origin</slot>
          </initarg>
          <initarg name="destination">
            <slot>destination</slot>
          </initarg>
        </initargs>
      </child-spec>
      <child-spec>
        <type-ref>unload-step</type-ref>
        <initargs>
          <initarg name="place">
            <slot>destination</slot>
          </initarg>
        </initargs>
      </child-spec>
    </decomposition>
    <constraints>
      <constraint>
        <type-ref>successor</type-ref>
        <from-type>load-step</from-type>
        <to-type>move</to-type>
        <ub><interval><minutes>0</minutes></interval></ub>
      </constraint>
      <constraint>
        <type-ref>successor</type-ref>
        <from-type>move</from-type>
        <to-type>unload-step</to-type>
        <ub><interval><minutes>0</minutes></interval></ub>
      </constraint>
    </constraints>
  </move>
</type>
```

Figure 43: Structure and XML specification of the `airlift` activity type

5.2.1.3 Airdrop

The `airdrop` activity type, summarized in Figure 44, defines an activity network nearly identical to that of `airlift`, except that the `unload-step` is removed. The `airdrop` activity models a *fast-rope* style of aerial movement where personnel and/or cargo are deplaned while an aircraft is still airborne.[23]



```
<type>
  <move name="airdrop">
    <decomposition>
      <child-spec>
        <type-ref>load-step</type-ref>
        <initargs>
          <initarg name="place"><slot>origin</slot></initarg>
        </initargs>
      </child-spec>
      <child-spec>
        <type-ref>move</type-ref>
        <initargs>
          <initarg name="origin">
            <slot>origin</slot>
          </initarg>
          <initarg name="destination">
            <slot>destination</slot>
          </initarg>
        </initargs>
      </child-spec>
    </decomposition>
    <constraints>
      <constraint>
        <type-ref>successor</type-ref>
        <from-type>load-step</from-type>
        <to-type>move</to-type>
        <ub><interval><minutes>0</minutes></interval></ub>
      </constraint>
    </constraints>
  </move>
</type>
```

Figure 44: Structure and XML specification of the `airdrop` activity type

[23]The term *fast-rope* describes a rapelling method for deploying troops and equipment by dropping them down ropes that hang from low-flying rotor aircraft.

5.2.1.4 Event-w-Resource-Sourcing

The `event-w-sourcing` activity type is an `event` with a `RESOURCE-SOURCING` wrapper (from Figure 39) that is used to represent a patrol or support mission that involves a stationary orbit around a fixed location, in which case it can basically be treated as an event occurring at a single place. The `RESOURCE-SOURCING` allows the resource to be sourced from elsewhere. The XML specification for `event-w-sourcing` is presented in Figure 45.

```
<type>
  <event name="event-w-sourcing">
    <variables>resource-origin resource-destination</variables>
    <wrappers>
      <wrapper name="resource-sourcing">
        <initargs>
          <initarg name="resource-origin">
            <slot>resource-origin</slot>
          </initarg>
          <initarg name="resource-destination">
            <slot>resource-destination</slot>
          </initarg>
          <initarg name="origin"><slot>place</slot></initarg>
          <initarg name="destination"><slot>place</slot></initarg>
        </initargs>
      </wrapper>
    </wrappers>
  </event>
</type>
```

Figure 45: XML specification of the `event-w-sourcing` activity type

5.2.2 Activity Instances

Figure 46 presents a sampling of XML specifications for the **Embassy Rescue** plan of activity instances (on the right) and the constraint instances (on the left) that link them together.

5.2.3 Resource Types

In addition to the `MH-47` and `MH-60` aircraft described in Section 5.1.2 (**Resource Types**), the **Embassy Rescue** scenario makes use of three additional aircraft types: the `AC-130U`, `MC-130H` and `C-141`. It also uses the `working-MOG` component resource class, as indicated by the `load-step` and `unload-step` activity types shown in Figure 42. The XML specifications for each of these four resource types are shown in Figure 47.

5.2.4 Capabilities

The capabilities used in the **Embassy Rescue** scenario are presented in Figure 48.

5.2.5 Resource Instances

The resource instances available for the **Embassy Rescue** scenario are summarized in Table 9. The left side of the figure lists all of the mobile resources in the standard **Embassy Rescue** situation, indicating how many are available and at what initial location (i.e., place). On the right side of the figure, the place components (i.e., those providing `working-MOG` capacity) are identified. Note that three places: `home-station`, `local-airport`, and `staging-area` each provide unlimited `working-MOG` capacity, while the `embassy` provides only two units of capacity, indicating that the processing of aircraft within the `embassy` is limited to just two aircraft at a time.

Table 9: `Embassy-Rescue` resource instances

Resource Type	Number of Instances	Initial Location	Place	Component Type	Capacity
AC-130U	1	home-station	air-corridor		
C-141	2	home-station	bridge		
FOOT	∞	n/a	embassy	working-MOG	2
MC-130H	5	home-station	home-station	working-MOG	∞
MH-47	5	home-station	local-airport	working-MOG	∞
MH-60	7	home-station	staging-area	working-MOG	∞

Deploy to Airport

```
<instance>
  <activity name="deploy-to-airport">
    <type-ref>airdrop</type-ref>
    <description>
      Deploy Taskforce BRAVO to airport
    </description>
    <origin>staging-area</origin>
    <destination>local-airport</destination>
    <wrappers>
      <wrapper name="resource-sourcing">
        <initargs>
          <initarg name="origin"><slot>origin</slot></initarg>
          <initarg name="destination">
            <slot>destination</slot>
          </initarg>
        </initargs>
      </wrapper>
    </wrappers>
    <manifest>
      <manifest-entry>
        <cargo count="64">PAX</cargo>
      </manifest-entry>
    </manifest>
    <requirements>
      <requirement>
        <capabilities>air-drop</capabilities>
      </requirement>
    </requirements>
  </activity>
</instance>
```

```
<instance>
  <constraint>
    <type-ref>before</type-ref>
    <from>deploy-to-airport</from>
    <to>secure-airport</to>
  </constraint>
</instance>
```

Secure Airport

```
<instance>
  <activity name="secure-airport">
    <type-ref>event</type-ref>
    <description>
      Establish control of local-airport
    </description>
    <place>local-airport</place>
    <duration><interval><hours>2</hours></interval></duration>
  </activity>
</instance>
```

```
<instance>
  <constraint>
    <type-ref>before</type-ref>
    <from>secure-airport</from>
    <to>
      evacuate-embassy
      retreat-to-airport
      prepare-for-extraction
    </to>
  </constraint>
</instance>
```

Evacuate Embassy

```
<instance>
  <activity name="evacuate-embassy">
    <type-ref>airlift</type-ref>
    <description>
      transport AmCits to local-airport for extraction
    </description>
    <origin>embassy</origin>
    <destination>local-airport</destination>
    <wrappers>
      <wrapper name="resource-sourcing">
        <initargs>
          <initarg name="origin"><slot>origin</slot></initarg>
          <initarg name="destination">
            <slot>destination</slot>
          </initarg>
        </initargs>
      </wrapper>
    </wrappers>
    <requirements>
      <requirement>
        <capabilities>light-transport</capabilities>
      </requirement>
    </requirements>
    <manifest>
      <manifest-entry>
        <cargo count="274">PAX</cargo>
      </manifest-entry>
    </manifest>
  </activity>
</instance>
```

Figure 46: XML specifications for select Embassy-Rescue activity and constraint instances

```
<type>
  <aircraft name="AC-130U">
    <description>
      Gunship: close air support,
      air interdiction, and force protection
    </description>
    <model-type>FIXED-WING</model-type>
    <speed units="mph">374</speed>
    <configurations>
      <configuration name=":default" op="or">
        <cargo max="40">PAX</cargo>
        <cargo max="1">ARMS</cargo>
      </configuration>
      <configuration name="close-air-support">
        <cargo max="5">MUNITIONS</cargo>
      </configuration>
    </configurations>
  </aircraft>
</type>

<type>
  <aircraft name="C-141">
    <description>
      Starlifter: cargo and troop support
    </description>
    <model-type>FIXED-WING</model-type>
    <speed units="mph">500</speed>
    <configurations>
      <configuration name=":default" op="or">
        <cargo max="200">PAX</cargo>
        <cargo max="6">PALLETS</cargo>
      </configuration>
      <configuration name="heavy-transport">
        <cargo max="200">PAX</cargo>
      </configuration>
    </configurations>
  </aircraft>
</type>
```

```
<type>
  <aircraft name="MC-130H">
    <description>
      Combat Talon II: infiltration,
      exfiltration, and resupply of
      special operations forces
    </description>
    <model-type>FIXED-WING</model-type>
    <speed units="mph">374</speed>
    <configurations>
      <configuration name=":default" op="or">
        <cargo max="92">PAX</cargo>
        <cargo max="5">PALLETS</cargo>
      </configuration>
      <configuration name="heavy-transport">
        <cargo max="92">PAX</cargo>
      </configuration>
      <configuration name="air-drop">
        <cargo max="64">PAX</cargo>
      </configuration>
    </configurations>
  </aircraft>
</type>


<type>
  <component name="working-MOG">
    <description>
      Workspace for loading, offloading,
      and servicing aircraft
    </description>
    <configurations>
      <configuration name=":default">
        <cargo max="1">AIRCRAFT</cargo>
      <configuration>
    </configurations>
  </component>
</type>
```

Figure 47: XML specifications of Embassy-Rescue resource types

```
<capability name="heavy-transport">
  <capable-types>MC-130H C-141</capable-types>
  <type-ref>move</type-ref>
</capability>

<capability name="light-transport">
  <capable-types>MH-60 MH-47</capable-types>
  <type-ref>move</type-ref>
</capability>

<capability name="air-drop">
  <capable-types>MH-60 MC-130H</capable-types>
  <type-ref>move</type-ref>
</capability>

<capability name="close-air-support">
  <capable-types>AC-130U MH-60</capable-types>
  <type-ref>move</type-ref>
</capability>
```

Figure 48: XML specifications of `Embassy-Rescue` capabilities

5.3 Exotic Dancer

The **Exotic Dancer** scenario represents the other end of the spectrum when compared to **Embassy Rescue**, but it more closely reflects the look and feel of SOF planning/scheduling problems, owing to its generation by the SOFTools application program. As mentioned earlier, the **Exotic Dancer** scenario involves the assault on two separate remote targets. The assault on `Target X` begins with an aircraft positioning leg from the home base (i.e., `Tampa`) to a staging base (i.e., `FSB Black`), and then an infiltration leg using `MH-47s` to carry troops and cargo in the form of 48 `PAX` and four `HMMWVs`. At an in-theater landing zone, the troops and cargo are dropped, and the attack on `Target X` proceeds using the `HMMWVs`. Following the attack, exfiltration reverses the steps, using the same resources. During the movements from `FSB Black` to `Target X`, three synchronized support missions are also executed: `Escort & Fire Support`, `Combat Search & Rescue`, and `Fire Support` (the latter encompassing the final push to `Target X`).

The second half of the scenario involves an independent water-based assault on `Target Y`, which begins from a ship at sea (i.e., `AFSB White`). The infiltration towards `Target Y` involves two steps, the first using two `MK-V` boats, which drop their eight troops into two `CRRC` rafts for a beach landing at `BLS Orange`. The final movement to `Target Y` takes place on foot. Following the assault, the exfiltration again reverses the infiltration steps, and the troops return to `AFSB White`. Beginning with the initial movement from `AFSB White`, an additional `Combat Search & Rescue` support mission is run, and a `Fire Support` mission executes during the attack on `Target Y`.

The complete **Exotic Dancer** scenario is summarized in Figure 49, and the entire plan, as represented within COMIREM, is presented in the subsequent three figures. Figure 50 shows the initial positioning leg from `Tampa` to `FSB Black` and the extent of the separate attacks on `Target X` and `Target Y`. Figure 51 shows the full attack on `Target X`, while Figure 52 shows the full attack on `Target Y`.

It is important to reiterate the following points about the **Exotic Dancer** scenario:

1. It does not include any explicit temporal sequencing constraints. Each activity in the plan is essentially fixed in time (the start and end times are all linked to one of two ref-hours for the plan: `H` and `N`). The temporal relations between all activities, including those that follow one another, and those, like the support missions, that encompass others, are all implicit.

2. It specifies exactly which resources, and precisely how many of each, are required by each activity in the plan (i.e., COMIREM's capability mechanism is not used). COMIREM ignores its own calculations based on manifest quantities and resource configurations, and instead relies on the numbers provided in the plan (specified using `count` attributes in the resource requirements) to determine how many resources to allocate to each activity.

The **Exotic Dancer** scenario illustrates how COMIREM can be used as a feasibility-checking mechanism for plans generated within a SOFTools-like program, where a planner with significant domain knowledge is able to specify a plan that indicates, in detail, what activities will be executed, and which particular resources will be used. COMIREM can then be called to determine whether the indicated resources are indeed available in sufficient quantity and at the proper lo-

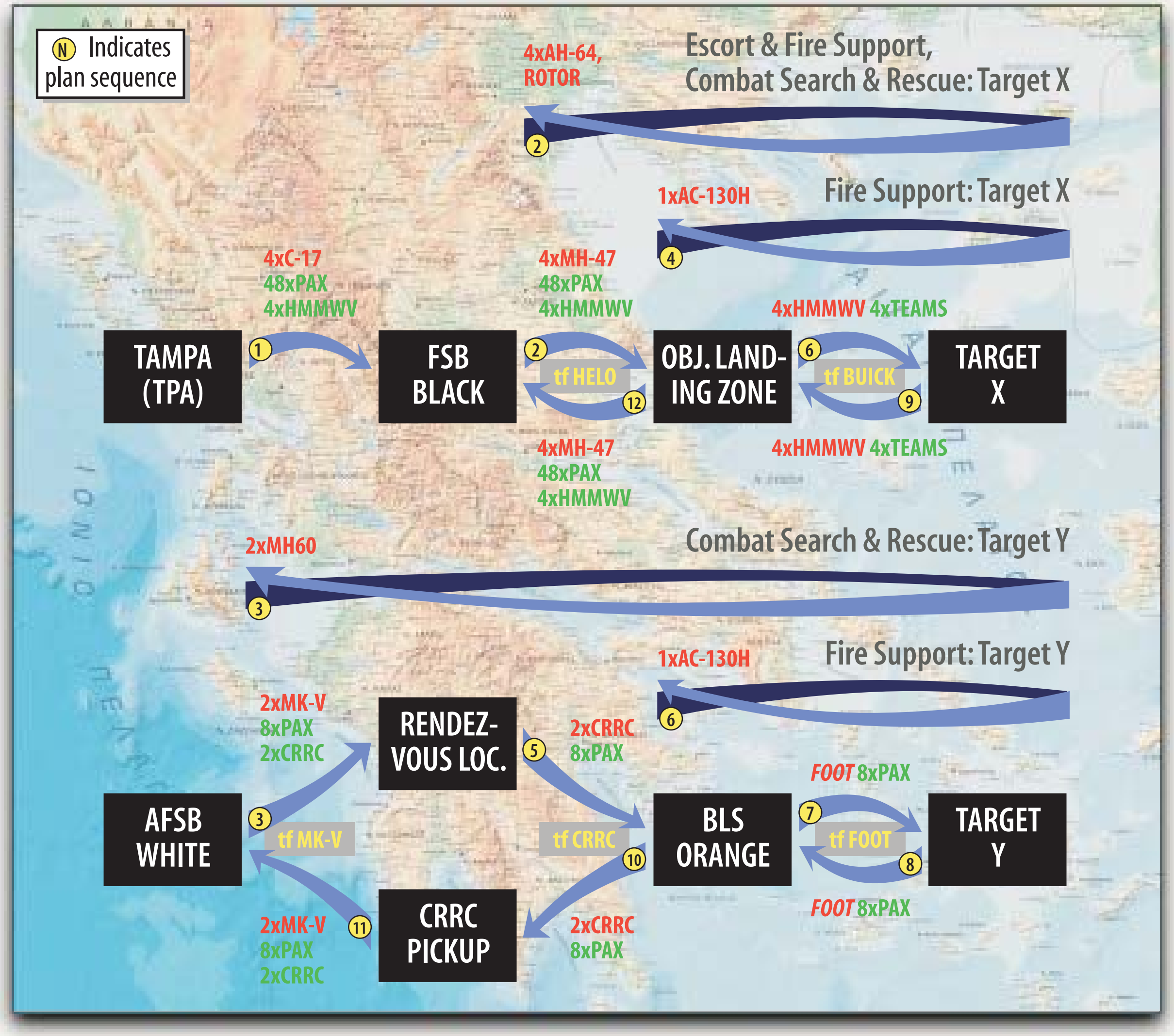


Figure 49: Exotic Dancer scenario

cations to satisfy the intentions of the planner. In the event that an infeasibility is encountered, the planner has the option of modifying and resubmitting the plan.

An example of this problem occurs in the **Exotic Dancer** plan as it currently exists. In the attack on `Target Y`, the second leg of the approach by sea using the two CRRC rafts is initially allotted—by the planner—a duration of 52 minutes. Given the maximum traveling speed of the CRRC rafts, which according to the domain model is 10 mph, and the required traveling distance from the `Rendezvous Location` to `BLS Orange`, which is 32 miles, COMIREM determines that the movement will actually require 192 minutes of travel time using the CRRCs, and therefore that there is not enough time for this activity to complete within the planner's bounds.

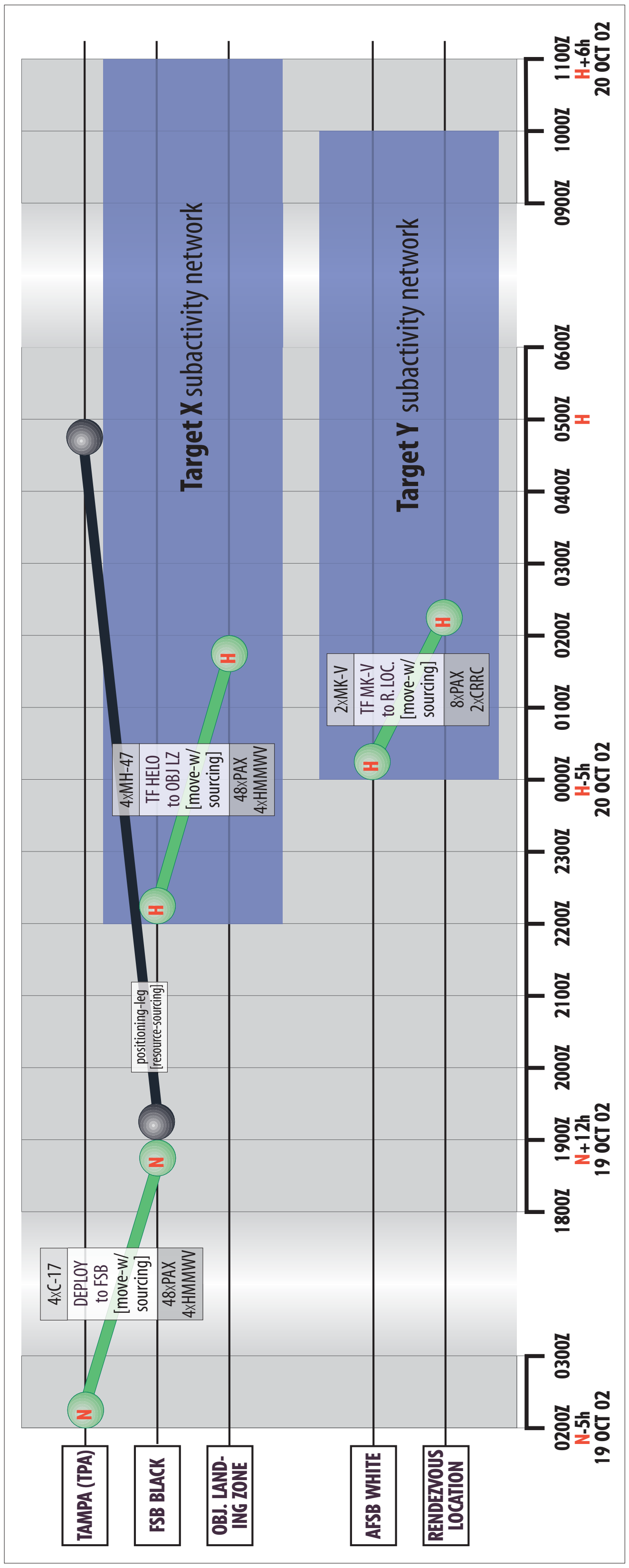


Figure 50: Exotic-Dancer activity network (summary view)

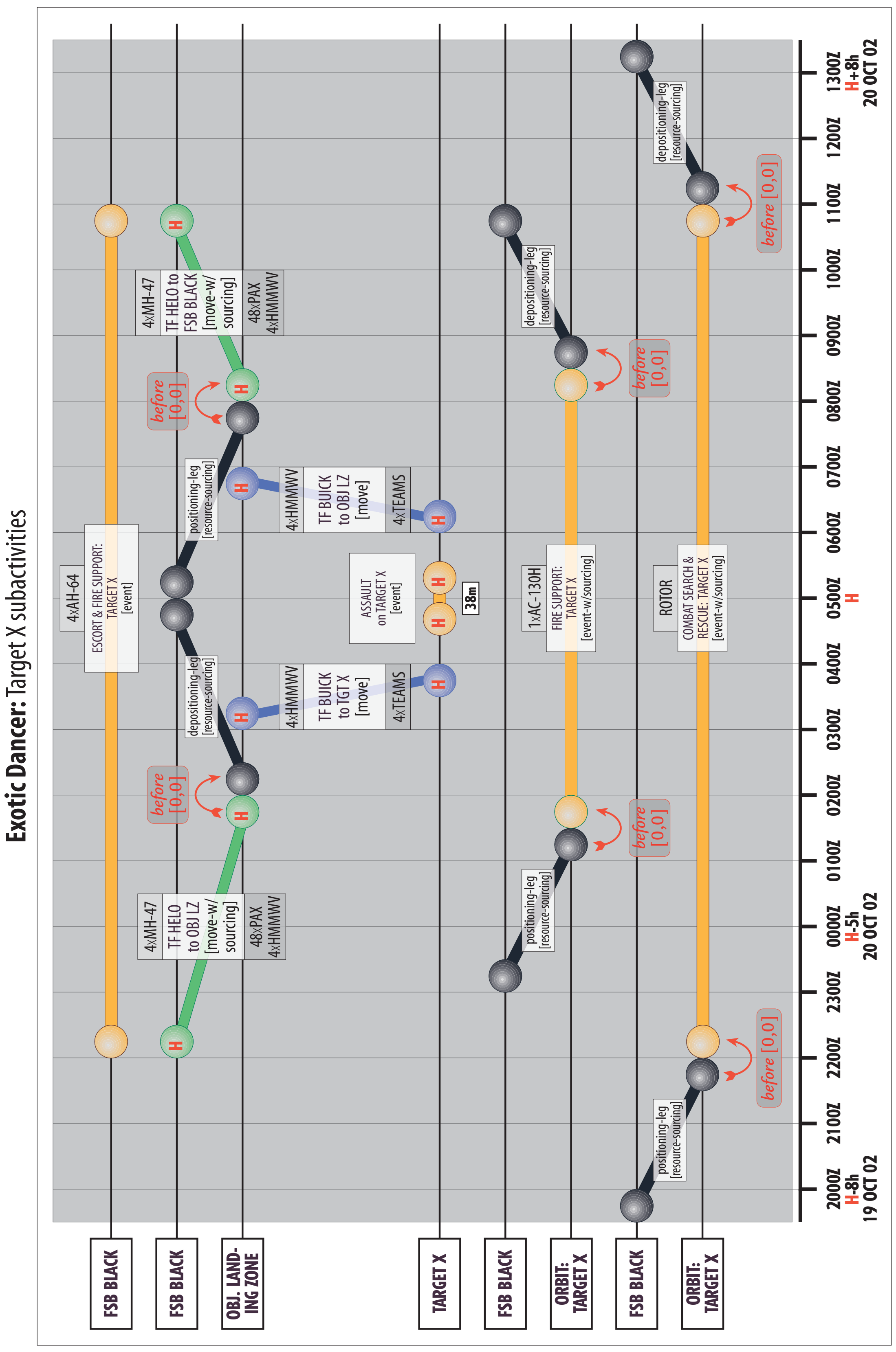

Exotic Dancer: Target X subactivities
FSB BLACK
FSB BLACK
OBJ. LANDING ZONE
TARGET X
FSB BLACK
ORBIT: TARGET X
FSB BLACK
ORBIT: TARGET X
4xAH-64
ESCORT & FIRE SUPPORT: TARGET X [event]
4xMH-47
TF HELO to OBJ LZ [move-w/ sourcing]
48xPAX 4xHMMWV
4xMH-47
TF HELO to FSB BLACK [move-w/ sourcing]
48xPAX 4xHMMWV
before [0,0]
before [0,0]
depositioning-leg [resource-sourcing]
positioning-leg [resource-sourcing]
4xHMMWV
TF BUICK to TGT X [move]
4xTEAMS
4xHMMWV
TF BUICK to OBJ LZ [move]
4xTEAMS
ASSAULT on TARGET X [event]
38m
H
positioning-leg [resource-sourcing]
depositioning-leg [resource-sourcing]
1xAC-130H
FIRE SUPPORT: TARGET X [event-w/sourcing]
before [0,0]
before [0,0]
ROTOR
COMBAT SEARCH & RESCUE: TARGET X [event-w/sourcing]
positioning-leg [resource-sourcing]
depositioning-leg [resource-sourcing]
before [0,0]
before [0,0]
2000Z H-8h 19 OCT 02
2100Z
2200Z
2300Z
0000Z H-5h 20 OCT 02
0100Z
0200Z
0300Z
0400Z
0500Z H
0600Z
0700Z
0800Z
0900Z
1000Z
1100Z
1200Z
1300Z H+8h 20 OCT 02

Figure 51: `Exotic-Dancer` activity subnetwork (`Target X` tasks)

Exotic Dancer: Target Y subactivities

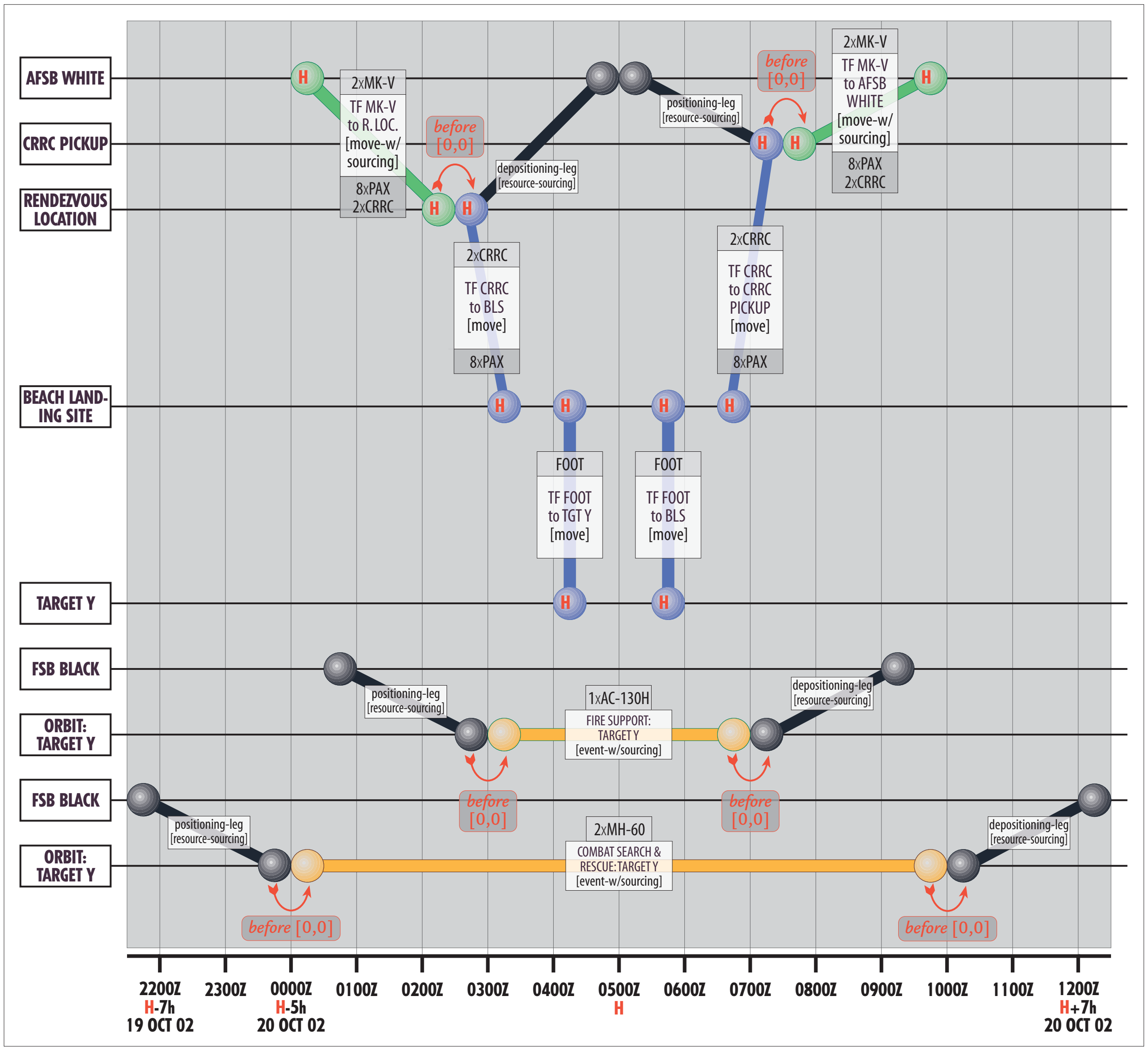


Figure 52: Exotic-Dancer activity subnetwork (Target Y tasks)

The planner has two sets of options at this point, relying on either modification of the plan by hand, or the conflict-resolution functionality provided by COMIREM, as described in Section 3.4.3 (**Conflict Analysis and User-Oriented Explanation**). The plan-modification options are as follows:

- Modify the plan by changing the amount of time allotted to the movement involving the CRRC rafts, to allow enough time to travel the 32 miles from the `Rendezvous Location` to `BLS Orange`.
- Modify the plan by changing the specific locations of the `Rendezvous Location` and/or `BLS Orange`, to shorten the distance between the two places.
- Modify the plan by attaching a different resource requirement to the activity.

The COMIREM conflict-resolution options are:

- Override the domain model to accept the viability of the shorter duration, under the assumption that special conditions or other situational knowledge justify it.
- Accept the longer duration determined by COMIREM, including any propagated temporal changes. Note that this is a case where the lack of any explicit temporal sequencing constraints can be a problem. COMIREM has no information with which to assess additional conflicts that may result from this kind of relaxation. As a result, it becomes the responsibility of the planner to determine the impact of this action.
- Determine whether any faster resources are available to be allocated to this activity, and offer them as alternative scheduling options to the user.

Regardless of which option the user ultimately takes, COMIREM has provided a valuable service in assessing the resource-feasibility of the plan as specified and offering a set of viable conflict-resolution options.

5.3.1 Activity Types

The only activity type utilized in the **Exotic Dancer** scenario is the `RESOURCE-SOURCING` wrapper (described in Section 5.1.1, **Activity Types**).

5.3.2 Activity Instances

Figure 53 presents a sampling of XML specifications of **Exotic Dancer** plan activity instances. Note the use of `count` attributes in the `requirements` tags to dictate how many resource instances are required for each task, and the explicit sourcing locations sent to the `RESOURCE-SOURCING` wrapper activity by the `FSB-Deployment` (top) and `CSAR-Target-Y` (bottom) activities.

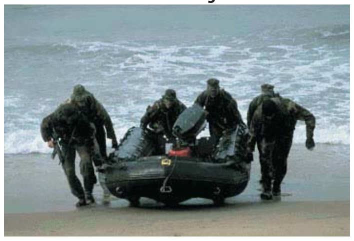
5.3.3 Resource Types

The **Exotic Dancer** scenario utilizes the resource types listed in Figure 40, Section 5.1.2 (**Resource Types**), page 65, and the additional resource types defined in Figure 54.

FSB Deployment (from TPA)

```
<instance>
  <activity name="M614077">
    <display-name>FSB-Deployment</display-name>
    <type-ref>move</type-ref>
    <description>
      Deployment from TPA to FSB-Black for Target-X assault
    </description>
    <origin>P852062</origin>
    <destination>P441277</destination>
    <wrappers>
      <wrapper name="resource-sourcing">
        <initargs>
          <initarg name="resource-origin"><value>P852062</value></initarg>
          <initarg name="resource-destination"><value>P852062</value></initarg>
          <initarg name="origin"><slot>origin</slot></initarg>
          <initarg name="destination"><slot>destination</slot></initarg>
        </initargs>
      </wrapper>
    </wrappers>
    <requirements>
      <requirement>
        <resource-class count="4">C-17</resource-class>
      </requirement>
    </requirements>
    <time-bounds-constraints>
      <ref-hour name="N" type="start">
        1034992848
      </ref-hour>
      <ref-hour name="N" type="finish">
        1035055152
      </ref-hour>
    </time-bounds-constraints>
  </activity>
</instance>
```

TF-CRRC: Landing at BLS



```
<instance>
  <activity name="M901480">
    <display-name>TF-CRRC:BLS-Landing</display-name>
    <type-ref>move</type-ref>
    <description>
      TF-CRRC arrival at BLS-Orange for Target-X assault
    </description>
    <origin>P152969</origin>
    <destination>P484554</destination>
    <requirements>
      <requirement>
        <resource-class count="2">CRRC</resource-class>
      </requirement>
    </requirements>
    <manifest>
      <manifest-entry>
        <cargo count"8">PAX</cargo>
      </manifest-entry>
    </manifest>
    <time-bounds-constraints>
      <ref-hour name="H" type="start">
        1035080990
      </ref-hour>
      <ref-hour name="H" type="finish">
        1035084494
      </ref-hour>
    </time-bounds-constraints>
  </activity>
</instance>
```

Combat Search & Rescue: Target-Y

```
<instance>
  <activity name="M18300-M951819">
    <display-name>CSAR-Target-Y</display-name>
    <type-ref>event</type-ref>
    <capability-type>move</capability-type>
    <description>
      Combat Search and Rescue for Target-Y
    </description>
    <place>P255811</place>
    <wrappers>
      <wrapper name="resource-sourcing">
        <initargs>
          <initarg name="resource-origin"><value>P569440</value></initarg>
          <initarg name="resource-destination"><value>P569440</value></initarg>
          <initarg name="origin"><slot>place</slot></initarg>
          <initarg name="destination"><slot>place</slot></initarg>
         </initargs>
      </wrapper>
    </wrappers>
    <requirements>
      <requirement>
        <resource-class count="2">MH-60</resource-class>
      </requirement>
    </requirements>
    <time-bounds-constraints>
      <ref-hour name="H" type="start">
        1035075079
      </ref-hour>
      <ref-hour name="H" type="finish">
        1035123119
      </ref-hour>
    </time-bounds-constraints>
    <duration>
      <interval><seconds>48040</seconds></interval>
    </duration>
  </activity>
</instance>
```

Figure 53: XML specifications for select `Exotic-Dancer` activity instances

```
<type>
  <aircraft name="AC-130H">
    <description>
      Gunship: close air support, air
      interdiction, and force protection
    </description>
    <model-type>FIXED-WING</model-type>
    <speed units="mph">374</speed>
    <configurations>
      <configuration name=":default">
        <cargo max="10">MUNITIONS</cargo>
      </configuration>
   </configurations>
  </aircraft>
</type>

<type>
  <aircraft name="AH-64">
    <description>
      Apache: attack helicopter
    </description>
    <model-type>FIXED-WING</model-type>
    <speed units="mph">374</speed>
    <configurations>
      <configuration name=":default">
        <cargo max="5">MUNITIONS</cargo>
      </configuration>
    </configurations>
  </aircraft>
</type>

<type>
  <aircraft name="C-17">
    <description>
      Globemaster III: cargo and
      troop transport
    </description>
    <model-type>FIXED-WING</model-type>
    <speed units="mph">500</speed>
    <configurations>
      <configuration name=":default">
        <cargo max="200">PAX</cargo>
      </configuration>
    </configurations>
  </aircraft>
</type>
```

```
<type>
  <seacraft name="CRRC">
    <description>
      Combat Rubber Raiding Craft:
      light SOF insertion/exfiltration
    </description>
    <model-type>POWERED</model-type>
    <speed units="mph">10</speed>
    <configurations>
      <configuration name=":default">
        <cargo max="8">PAX</cargo>
      </configuration>
    </configurations>
  </seacraft>
</type>

<type>
  <seacraft name="MK-V">
    <description>
      Special Operations Craft:
      cargo and SOF support
    </description>
    <model-type>POWERED</model-type>
    <speed units="mph">70</speed>
    <configurations>
      <configuration name=":default">
        <cargo max="20">PAX</cargo>
      </configuration>
    </configurations>
  </seacraft>
</type>

<type>
  <landcraft name="HMMWV">
    <description>
      High-Mobility Multipurpose
      Vehicle: light tactical vehicle
    </description>
    <model-type>WHEELED</model-type>
    <speed units="mph">10</speed>
    <configurations>
      <configuration name=":default">
        <cargo max="8">PAX</cargo>
      </configuration>
    </configurations>
  </landcraft>
</type>
```

top (l-r):
AC-130H
AH-64
C-17

bottom:
CRRC
MK-V
HMMWV

Figure 54: XML specifications of Exotic-Dancer resource types

5.3.4 Resource Instances

All of the place instances in **Exotic Dancer** exist solely to serve as named locations in the plan (i.e., for calculating travel distances), which is to say, there are no accompanying component resources (like `working-MOG`). The mobile resources are also apportioned in such a way as to facilitate a feasible solution. For completeness' sake, a description of the **Exotic Dancer** resource instances is included in Table 10.

Table 10: `Exotic-Dancer` resource instances

	Resource Type	Number of Instances	Initial Location
Aircraft	AC-130H	1	Tampa (TPA)
		2	FSB Black
	AH-64	4	FSB Black
	C-17	4	Tampa (TPA)
	MH-47	5	FSB Black
	MH-60	3	FSB Black
Seacraft	CRRC	2	Rendezvous Location
	MK-V	2	AFSB White
Landcraft	HMMWV	4	Objective Landing Zone
Organic	Infantry	20	Tampa (TPA)
	FOOT	∞	n/a

definition of multiple configurations for a resource, an extension that would allow it to instantiate derivative setup operations to perform necessary configuration changes (with situation-specific processing durations) would require very little in the way of additional mechanisms and make COMIREM more widely applicable to manufacturing-like problem domains.

6.1.7 Abstract Resource Allocation

Assignment of resources ought to be possible at any level in the COMIREM resource-requirement hierarchy (see Figure 16, Section 3.2.2.1 (**The Resource Requirement Hierarchy**), page 29). For example, it should be possible to commit to a `ROTOR` aircraft for a given activity and later refine this choice to a specific helicopter type (i.e., `MH-60`). The system should also be capable of reasoning with these abstract partial assignments. That is, if a (partial) assignment is made at the level of a basic air capability (e.g., an ISR asset) and there are no ISR assets available over the period of time during which it is needed, then the system should recognize this problem. Propagation of resource constraints in this example must be capable of isolating the set of instances that could provide this capability and confirming availability of at least one instance. More generally, the system should be capable of maintaining and reasoning with a plan/schedule that contains resource assignments or partial assignments at different levels of specificity. Going one step further, it should be possible for the system to recognize situations where user decisions have sufficiently constrained remaining choices such that only one unique alternative exists at some level. In these cases the system should proceed to make this unique assignment—which could still be a partial one, and which the interface should make clear is a derived choice.

6.2 Scheduling Issues

A great opportunity exists in COMIREM for enhancing the scheduling engine by opening it up to accepting additional input from the user. We are interested generally in the design of user-directed strategies for flexible-times scheduling, including allowing the user to specify relaxable constraints and objective criteria to drive the scheduler. Users ought to be able to specify preferences for the tightness of various constraints during the scheduling process, and dictate high-level goals for the overall schedule (or perhaps just portions of the schedule), such as lowest risk, lowest cost or shortest makespan.

We are also currently in the process of investigating extensions to facilitate explicit goal-directed reasoning about the resource-support plans required to enable various resource assignments (see [Smith and Zimmerman, 2004]).

6.3 Planning Issues

A major area of continuing interest for COMIREM is the enhancement of its planning capabilities. We envision providing direct interactive support for creating and manipulating activities and activity networks through the GUI to extend COMIREM's forms-based mixed-initiative approach further into the realm of planning.

Users ought to be able to create activity networks (perhaps in a fashion similar to that of SOFTools/SOFPlan) and augment/modify existing networks using active-forms-style GUI templates. Moves can be defined by establishing temporal links (i.e., using a drag-and-drop opera-

```
<requirements>
   <requirement><resource-class> R1 </resource-class></requirement>
   <requirement><resource-class> R2 </resource-class></requirement>
   <requirement><resource-class> R3 </resource-class></requirement>
</requirements>
```

the assumption of the scheduler is that an instance of *each of* resource class $\mathbf{R}_1$, $\mathbf{R}_2$ *and* $\mathbf{R}_3$ must be reserved for the duration of the activity. One obvious extension to this format would be to add a logical operator attribute to the `<requirements>` tag that would indicate how the constituent requirements are to be interpreted (i.e., using either an *and* or *or*). In the case of an *or* operator, as in `<requirements operator="or">...</requirements>`, the scheduler would need only reserve an instance of one of the $\mathbf{R}_n$ to satisfy the needs of the activity. Further extensions could be made to support an even richer resource-requirement logic, using nested combinations of *and* and *or* operators.[24]

6.1.4 Consumable Resources

Consumable commodities (e.g., fuel, munitions, money) are not currently supported by COMIREM. The work of Laborie (see above) again offers useful techniques for representing and managing consumable resources that could be used to extend the COMIREM resource model.

6.1.5 Producer and Consumer Relations

An additional class of resource-usage constraints not yet handled by COMIREM is the producer-consumer relation. For example, in transshipment contexts, where cargos are being moved to intermediate locations and then reconstituted for subsequent movement elsewhere, the relative carrying capacities of the resources involved at each stage may allow execution of respective activities to be overlapped, thereby reducing the overall movement duration. By adding some additional constraints to govern the repackaging of cargo, the sequencing of multiple-wave activities can be controlled at a much finer level of detail.

Producer and consumer relations can also play an important role in more heavily planning-related situations, like those involving the development of a product, as in an ISR, manufacturing or workflow environment. Sequencing constraints defined as producer-consumer links allow a system to respond to execution results with greater flexibility. For example, when a product does not meet specifications, the system may be able to iteratively modify and/or reschedule the original production steps or seek other avenues for securing satisfactory product.

6.1.6 Situation-dependent Setup Operations

The concept of situation-dependent resource setup operations is a short logical step from that of derivative (de)positioning legs for mobile resources. Given that COMIREM already supports the

[24]It should be reiterated that the *model-type* in the resource-requirement hierarchy already provides a degree of *or* functionality by allowing multiple resource classes to be grouped together into a single class. For example, a reference to the `ROTOR` model-type in a resource requirement permits COMIREM to allocate either an `MH-47` or `MH-60` helicopter to an activity.

definition of multiple configurations for a resource, an extension that would allow it to instantiate derivative setup operations to perform necessary configuration changes (with situation-specific processing durations) would require very little in the way of additional mechanisms and make Comirem more widely applicable to manufacturing-like problem domains.

6.1.7 Abstract Resource Allocation

Assignment of resources ought to be possible at any level in the Comirem resource-requirement hierarchy (see Figure 16, Section 3.2.2.1 (**The Resource Requirement Hierarchy**), page 29). For example, it should be possible to commit to a `ROTOR` aircraft for a given activity and later refine this choice to a specific helicopter type (i.e., `MH-60`). The system should also be capable of reasoning with these abstract partial assignments. That is, if a (partial) assignment is made at the level of a basic air capability (e.g., an ISR asset) and there are no ISR assets available over the period of time during which it is needed, then the system should recognize this problem. Propagation of resource constraints in this example must be capable of isolating the set of instances that could provide this capability and confirming availability of at least one instance. More generally, the system should be capable of maintaining and reasoning with a plan/schedule that contains resource assignments or partial assignments at different levels of specificity. Going one step further, it should be possible for the system to recognize situations where user decisions have sufficiently constrained remaining choices such that only one unique alternative exists at some level. In these cases the system should proceed to make this unique assignment—which could still be a partial one, and which the interface should make clear is a derived choice.

6.2 Scheduling Issues

A great opportunity exists in Comirem for enhancing the scheduling engine by opening it up to accepting additional input from the user. We are interested generally in the design of user-directed strategies for flexible-times scheduling, including allowing the user to specify relaxable constraints and objective criteria to drive the scheduler. Users ought to be able to specify preferences for the tightness of various constraints during the scheduling process, and dictate high-level goals for the overall schedule (or perhaps just portions of the schedule), such as lowest risk, lowest cost or shortest makespan.

We are also currently in the process of investigating extensions to facilitate explicit goal-directed reasoning about the resource-support plans required to enable various resource assignments (see [Smith and Zimmerman, 2004]).

6.3 Planning Issues

A major area of continuing interest for Comirem is the enhancement of its planning capabilities. We envision providing direct interactive support for creating and manipulating activities and activity networks through the GUI to extend Comirem's forms-based mixed-initiative approach further into the realm of planning.

Users ought to be able to create activity networks (perhaps in a fashion similar to that of SOFTools/SOFPlan) and augment/modify existing networks using active-forms-style GUI templates. Moves can be defined by establishing temporal links (i.e., using a drag-and-drop operation) between two locations; events can be defined similarly by specifying a temporal interval at a single location. Sequencing constraints, similar to moves, are temporal links (perhaps annotated) between activities. Finally, the XML specification for an activity type ought to be derivable directly from existing instantiations, for future use in other plans.

6.4 Architectural Issues

COMIREM's lightweight client/server design provides the basis for a number of possible extensions towards supporting a fully multi-user planning and scheduling environment. As shown in Figure 55, a single COMIREM server could manage one or more planning/scheduling scenarios for multiple clients, with access controlled through locking and/or other mechanisms. Issues related to the synchronization of multiple modifications to the same scenario would need to be addressed formally or prevented outright. Additionally, the client/server design provides the means for supporting message passing between clients (e.g., users working on related scenarios or different parts of the same scenario) and even the dynamic modular configuration of planning/scheduling clients, where users would be able to tailor their own client environment based on their individual needs or context and thereby minimize even further the total footprint of their application and its impact on the server.

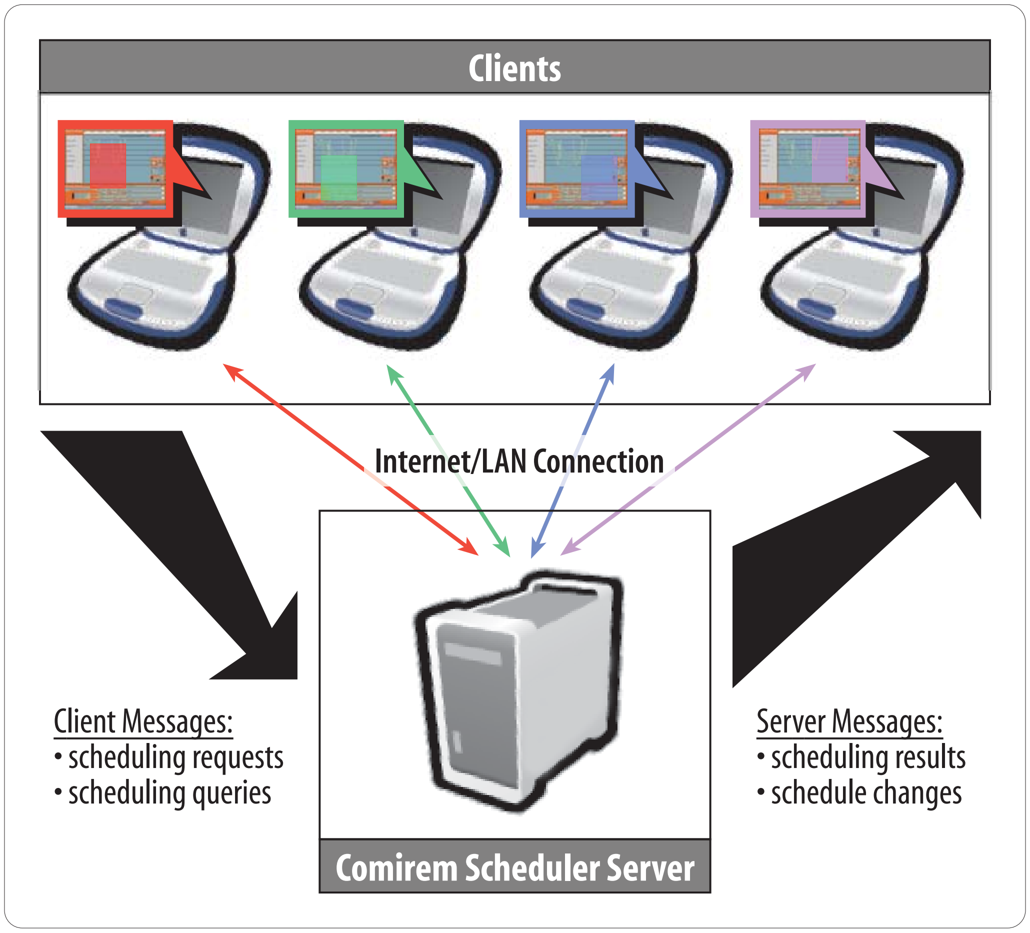


Figure 55: COMIREM multi-user client/server environment

References

[Becker and Smith, 2000] M.A. Becker and S.F. Smith. Mixed-initiative resource management: The AMC Barrel Allocator. In *Proceedings, Fifth International Conference on Artificial Intelligence Planning Systems*, pages 32–41, Breckenridge CO, April 2000. American Association for Artificial Intelligence (AAAI). Available via: *hyperlink* ⇝ PDF.

[Dechter *et al.*, 1991] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 41(1–3):61–95, May 1991. Special issue on knowledge representation.

[Harold and Means, 2002] Elliotte Rusty Harold and W. Scott Means. *XML In a Nutshell*. O'Reilly, Sebastopol CA, 2nd edition, June 2002.

[Keene, 1989] Sonya E. Keene. *Object-Oriented Programming in Common Lisp: A Programmer's Guide to CLOS*. Addison-Wesley, Reading MA, 1989.

[Laborie, 2003] Philippe Laborie. Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence*, 143(2):151–188, February 2003.

[Lafferty, 2001a] Larry Lafferty. Representing geographic data in the structured data model. Proposed standard for Active Templates program, 11 October 2001.

[Lafferty, 2001b] Larry Lafferty. Representing temporal data in the structured data model. Proposed standard for Active Templates program, 11 October 2001.

[Smith and Becker, 1997] S.F. Smith and M.A. Becker. An ontology for constructing scheduling systems. In *Working Notes, 1997 AAAI Spring Symposium on Ontological Engineering*, Stanford University, California, March 1997. AAAI Press. Available via: *hyperlink* ⇝ PDF.

[Smith and Zimmerman, 2004] S.F. Smith and T.L. Zimmerman. Planning tactics within scheduling problems. In *Proceedings, ICAPS-04 Workshop on Integrating Planning into Scheduling*, Whistler, Canada, June 2004.

[Smith *et al.*, 1996] S.F. Smith, O. Lassila, and M.A. Becker. Configurable, mixed-initiative systems for planning and scheduling. In A. Tate, editor, *Advanced Planning Technology*, pages 235–241. AAAI Press, Menlo Park CA, 1996 1996. Available via: *hyperlink* ⇝ PDF.

[Smith *et al.*, 2004] S.F. Smith, G. Cortellessa, D.W. Hildum, and C.M. Ohler. Using a scheduling domain ontology to compute user-oriented explanations. In *Proceedings, 16th European Conference on Artificial Intelligence Workshop on Planning and Scheduling: Bridging Theory to Practice*, Valencia, Spain, August 2004. Available via: *hyperlink* ⇝ PDF.

[Smith *et al.*, 2005] S.F. Smith, D.W. Hildum, and D.R. Crimm. Comirem: An intelligent form for resource management. *IEEE Intelligent Systems*, 2005. Special issue on Agent-Based Intelligent Forms, *To appear*.

[Steele Jr., 1990] Guy L. Steele Jr. *Common Lisp: The Language, Second Edition*. Digital Press, 1990.

A Input Data Specifications

This appendix provides the complete XML specification grammar for each of the data elements that can appear in a COMIREM input data file. It is organized into five sections, covering basic support entities (including constraints), activities (types and instances), resources (types and instances), capabilities, and plan configurations.

Figure 56 displays an XML-centered layout of the input data files that comprise a situation (i.e., the `types`, `capabilities` and `instances` files on the left) and scenario (i.e., the `plan` file on the right) in COMIREM, as introduced in Figure 1, Section 3.1 (**Representation of SOF Domains**), page 8.

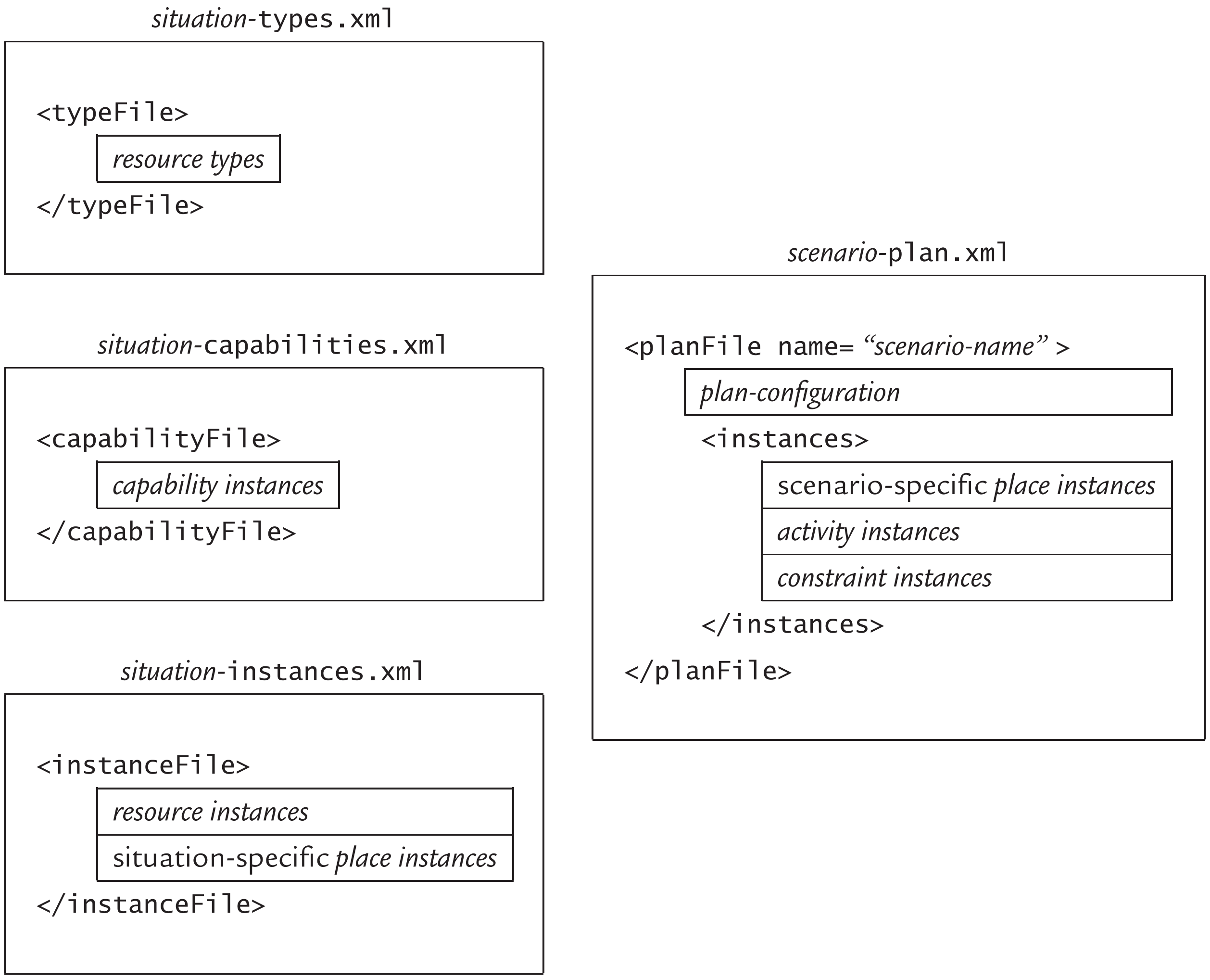


Figure 56: COMIREM input file format description

A.1 Supporting Entities

The supporting entities described in this section facilitate the specification of temporal, geographic, cargo and constraint information. These entities are referenced by multiple other

COMIREM data entities in the process of representing an input situation or scenario.

A.1.1 Time Formats

The time formats in this section match those described in Active Templates program document number 1: *Representing Temporal Data in the Structured Data Model* (see [Lafferty, 2001b]).

The basic temporal specifications in COMIREM cover both absolute times and time intervals. Absolute times are used for specifying release and due dates and constraining activity start and end times. Time intervals are used in the specification of upper and lower temporal bounds on both constraints and durations. A specific duration entity is provided for specifying minimum/maximum ranges and fixed time points for activity durations.

A.1.1.1 Absolute Time

Absolute time in COMIREM can be specified as a calendar date with an optional clock time (to thousandths of a second), either in Zulu time or offset from Zulu time by some number of hours and/or minutes. The time specification format is presented in Table 11.

Table 11: Time specification format

Grammar for *time-spec*			
time-spec	$\rightarrow$	*CCYYMMDD* 'T' *hhmmss* ['.'*sss*] [(*zulu-suffix* | *offset-suffix*)]	
		CC	the century component of a year; range: [00,99]
		YY	the tens and units components of a year; range: [00,99]
		MM	a month; range: [01,12]
		DD	a day; range: $[01, \left\{\begin{array}{l} 28 : \text{MM} = 02 \wedge \text{CCYY} \neq \text{leap year} \\ 29 : \text{MM} = 02 \wedge \text{CCYY} = \text{leap year} \\ 30 : \text{MM} \in \{04,06,09,11\} \\ 31 : \text{MM} \in \{01,03,05,07,08,10,12\} \end{array}\right\}]$
		hh	an hour; range: [00,23]
		mm	minutes; range: [00,59]
		ss	seconds; range: [00,59]
		sss	thousandths of a second; range: [000,999]
zulu-suffix	$\rightarrow$	'Z'	Designates Zulu time (the default is local)
offset-suffix	$\rightarrow$	('+' | '−') *hhmm*	Designates an offset from Zulu time

A.1.1.2 Intervals

Time intervals are blocks of time specified in terms of standard time units (e.g., seconds, minutes, hours), as well as the system-defined values: `:infinity` (i.e., an unbounded positive amount of time) and `:negative-infinity` (i.e., an unbounded negative amount of time).[25] The interval specification format is presented in Table 12.

Table 12: Interval specification format

Grammar for *interval-spec*		
interval-spec	→	`<interval>`
		time-spec | `:negative-infinity` | `:infinity`
		| (`<interval-string>` *interval-string* `</interval-string>`)
		`</interval>`
time-spec	→	[`<seconds>` INTEGER `</seconds>`] [`<minutes>` INTEGER `</minutes>`]
		[`<hours>` INTEGER `</hours>`] [`<days>` INTEGER `</days>`]
		[`<weeks>` INTEGER `</weeks>`] [`<months>` INTEGER `</months>`]
		[`<years>` INTEGER `</years>`]
interval-string	→	['–'] 'P' [INTEGER 'Y'] [INTEGER 'M'] [INTEGER 'D'] [*hms-suffix*]
hms-suffix	→	'T' [INTEGER 'H'] [INTEGER 'M'] [(INTEGER | FLOAT) 'S']

A.1.1.3 Duration

Durations in COMIREM can be specified as either fixed amounts of time or lower and/or upper-bounded time ranges (i.e., using the *interval-spec* time-interval format). The duration specification format is presented in Table 13.

A.1.2 Geographical Formats

The formats in this section handle such geographical entities as locations, distances and speeds.

A.1.2.1 Locations

The location formats in this section correspond to those described in Active Templates program document number 2: *Representing Geographic Data in the Structured Data Model* ([Lafferty, 2001a]).

Locations are represented by latitude/longitude pairs with an optional altitude value. In the absence of an altitude specification, a location is assumed to be at sea level. Latitude and longitude values are specified in degrees as floats using a `[-]DDD.ddd` format, with negative

[25]Note that the only way (currently) to specify a negative bounded time is to use the `interval-string` tag.

Table 13: Duration specification format

Grammar for *duration-spec*			Notes
duration-spec	$\rightarrow$	`<duration>` ([*min-spec*] [*max-spec*]) | *fixed-spec* `</duration>`	
min-spec	$\rightarrow$	`<minimum>` *interval-spec* `</minimum>`	**a**
max-spec	$\rightarrow$	`<maximum>` *interval-spec* `</maximum>`	**a**
fixed-spec	$\rightarrow$	`<fixed>` *interval-spec* `</fixed>`	**a**

a: for *interval-spec* grammar, see Table 12, page 91

values implying the southern hemisphere for latitude and the eastern hemisphere for longitude. Altitudes can be specified in a variety of distance units. The location specification format is presented in Table 14.

Table 14: Location specification format

Grammar for *location-spec*			Notes
location-spec	$\rightarrow$	`<location` *latitude-spec* *longitude-spec*`>` [*altitude-spec*] `</location>`	
latitude-spec	$\rightarrow$	`longitude`=“FLOAT”	
longitude-spec	$\rightarrow$	`latitude`=“FLOAT”	
altitude-spec	$\rightarrow$	`<altitude>` *distance-spec* `</altitude>`	**a**

a: for *distance-spec* grammar, see Table 15, page 93

A.1.2.2 Distance

Distances can be specified as integers or floats in units of feet, meters, yards, kilometers or miles. The distance specification format is presented in Table 15.

Table 15: Distance specification format

Grammar for *distance-spec*		
distance-spec	$\rightarrow$	`<distance units=`"*distance-units*"`>` INTEGER | FLOAT `</distance>`
distance-units	$\rightarrow$	`feet` | `meters` | `yards` | `kilometers` | `miles`

A.1.2.3 Speed

Speeds can be specified as integers or floats in units of feet-per-second (fps), miles-per-hour (mph), or knots. The speed specification format is presented in Table 16.

Table 16: Speed specification format

Grammar for *speed-spec*		
speed-spec	$\rightarrow$	`<speed units=`"*speed-units*"`>` INTEGER `</speed>`
speed-units	$\rightarrow$	`fps` | `mph` | `knots`

A.1.3 Cargo Format

The cargo format serves two purposes: it supports the specification of units of cargo comprising the manifest for an activity, and an upper bound on resource capacity when defining a resource configuration. The former case makes use of a `count` XML attribute, while the latter uses a `max` attribute. COMIREM provides a predefined set of generic cargo types: `PAX` (for personnel), `FUEL`, `MUNITIONS`, `PALLETS` and `AIRCRAFT`. The cargo specification format is presented in Table 17.

A.1.4 Constraint Formats

COMIREM constraint formats cover both temporal sequencing and reference hour constraints.

A.1.4.1 Temporal Sequencing Constraints

Temporal sequencing constraints are used to specify temporal relations among activities within a plan and among subactivities (i.e., children) of an aggregate activity. Each constraint must

Table 17: Cargo specification format

Grammar for *cargo-spec*		
cargo-spec	$\rightarrow$	`<cargo` [*count-spec*] [*max-spec*]`>` *cargo-type* `</cargo>`
count-spec	$\rightarrow$	`count=` "INTEGER"
max-spec	$\rightarrow$	`max=` "INTEGER"
cargo-type	$\rightarrow$	`PAX` | `FUEL` | `MUNITIONS` | `PALLETS` | `AIRCRAFT`

specify a set of nodes both *from* which the constraint relation emanates and *to* which it leads. When the constraint specifies a relation between subactivities, the *from-type* and *to-type* variants may be used, to take advantage of the fact that subactivities need not be named, and can therefore be accessed by their type alone—assuming that the type is unique among all of the (local) subactivities. Also in this case, the special keyword, `:child`, can be used to refer to a nexus activity from within a wrapper, since its name cannot be determined until instantiation time.

The predefined temporal constraint types, as defined in Section 3.2.3.1 (**Temporal Sequencing Constraints**), are: `successor/before`, `same-start` and `same-finish`, `overlaps` and `contains`. The temporal constraint specification format is presented in Table 18.

A.1.4.2 Reference Hour Constraints

Reference hour constraints are specified at the plan level and used to implement Comirem's ref-hours. They attach a name, say **P**, to a particular point in time fixed to Comirem's underlying constraint network's calendar zero, and may include an optional *delta* value that specifies a time window centered around **P**, i.e., [**P**$-\Delta$,**P**$+\Delta$], from which any time point linked to the ref-hour can be offset, using an additional value. The values of **P**, Δ and the offset can be changed at any time. If either of these values is modified, each of the time points linked to the ref-hour must be updated in response. The reference hour constraint specification format is presented in Table 19.

A.2 Activity Attributes

The various specification formats for activity attributes are presented in this section. Note that the distinction between type and instance attributes is not strictly defined (or enforced) in Comirem. That is to say, it is generally permissible to specify any attribute at either the type or instance level. Attributes specified at the type level will apply to all instances of the type, and attributes specified at the instance level will override type-level definitions. In this section, attributes have been grouped according to how they are typically specified in Comirem data files.

Table 18: Temporal sequencing constraint specification format

Grammar for *constraint-spec*			Notes
constraint-spec	→	`<constraint>` *type-ref-spec* *from-spec* *to-spec* [*lb-spec*] [*ub-spec*] `</constraint>`	
type-ref-spec	→	`<type-ref>` *constraint-type-name* `</type-ref>`	
constraint-type-name	→	`successor` | `before` | `same-start` | `same-finish` | `overlaps` | `contains`	
from-spec	→	`<from>` [`:child` | *activity-name*]+ `</from>`	**a**
		| `<from-type>` [*child-type-name*]+ `</from-type>` †	**b**
to-spec	→	`<to>` [`:child` | *activity-name*]+ `</to>`	**a**
		| `<to-type>` [*child-type-name*]+ `</to-type>` †	**b**
lb-spec	→	`<lb>` *interval-spec* `</lb>`	**c**
ub-spec	→	`<ub>` *interval-spec* `</ub>`	**c**

†: this tag is not relevant for plan-level constraints, i.e., in the *scenario*-`plan.xml` file

a: *activity-name* must uniquely match a current activity instance (the match must be with a child activity specified in an accompanying `decomposition` tag if the constraint defines an inter-activity relation)

b: *child-type-name* must uniquely match a child activity type as specified in an accompanying `decomposition` tag

c: for *interval-spec* grammar, see Table 12, page 91

As should be expected, activity type and instance names must be unique within a scenario. The activity type specification format is presented in Table 20, and the activity instance specification format is presented in Table 21.

Table 19: Reference hour constraint specification format

Grammar for *ref-hours-spec*			Notes
ref-hours-spec	$\rightarrow$	`<ref-hours>` [*ref-hour-spec*]+ `</ref-hours>`	
ref-hour-spec	$\rightarrow$	`<ref-hour name=`"*ref-hour-name*"`>`	
		`<time>` (*time-spec* | INTEGER) `</time>`	**a**
		`<delta>` *interval-spec* `</delta>`	**b**
		`</ref-hour>`	

a: for *time-spec* grammar, see Table 11, page 90
b: for *interval-spec* grammar, see Table 12, page 91

Table 20: Activity type specification format

Grammar for *activity-type-spec*			Notes
activity-type-spec	→	`<type>`	
		`<`*activity-type-name* `name=`"*new-type-name*"`>`	
		[*description-spec*]	
		[*variables-spec*]	
		[*inputs-spec*]	
		[*decomposition-spec*]	**a**
		[*constraints-spec*]	**b**
		[*wrappers-spec*]	**c**
		[*requirements-spec*]	**d**
		[`<derivative-p/>`]	
		[*capability-type-spec*]	
		`</`*activity-type-name*`>`	
		`</type>`	
activity-type-name	→	`wrapper` | `move` | `event`	
description-spec	→	`<description>` TEXT `</description>`	
variables-spec	→	`<variables>` [SYMBOL]+ `</variables>`	
inputs-spec	→	`<inputs>` [SYMBOL]+ `</inputs>`	
capability-type-spec	→	`<capability-type>` (`move` | `event`) `</capability-type>`	

a: for *decomposition-spec* grammar, see Table 22, page 99
b: for *constraints-spec* grammar, see Table 23, page 100
c: for *wrappers-spec* grammar, see Table 24, page 100
d: for *requirements-spec* grammar, see Table 26, page 102

Table 21: Activity instance specification format

Grammar for *activity-instance-spec*			Notes
activity-instance-spec	→	`<instance>`	
		`<activity name=`"*activity-name*"`>`	
		type-ref-spec [*display-name-spec*] [*time-constraints-spec*]	
		origin-spec *destination-spec* † *place-spec* ‡	
		[*initargs-spec*] [*manifest-spec*] [*duration-spec*]	**a**, **b**, **c**
		`</activity>`	
		`</instance>`	
type-ref-spec	→	`<type-ref>` *type-name* `</type-ref>`	**d**
display-name-spec	→	`<display-name>` TEXT `</display-name>`	
origin-spec	→	`<origin>` (*place-name* | *location-spec*) `</origin>`	**e**, **f**
destination-spec	→	`<destination>`	
		(*place-name* | *location-spec*)	**e**, **f**
		`</destination>`	
place-spec	→	`<place>` (*place-name* | *location-spec*) `</place>`	**e**, **f**
time-constraints-spec	→	`<time-bounds-constraints>`	
		[*ref-hour-con-spec* | *anchor-con-spec*]+	
		`</time-bounds-constraints>`	
ref-hour-con-spec	→	`<ref-hour name=`"*ref-hour-name*" `type=`"*time-point*"`>`	**g**
		(*time-spec* | INTEGER)	**h**
		`</ref-hour>`	
time-point	→	(`start` | `finish`)	
anchor-con-spec	→	`<anchor type=`"*time-constraint-type*"`>`	
		(*time-spec* | INTEGER)	**h**
		`</anchor>`	
time-constraint-type	→	EST | LST | EFT | LFT	

†: this option is only relevant for `move` activity types
‡: this option is only relevant for `event` activity types

a: for *initargs-spec* grammar, see Table 25, page 101
b: for *manifest-spec* grammar, see Table 27, page 103
c: for *duration-spec* grammar, see Table 13, page 92
d: *type-name* must refer to a predefined activity type
e: *place-name* must refer to a defined `place` resource instance
f: for *location-spec* grammar, see Table 14, page 92
g: *ref-hour-name* must refer to a predefined ref-hour
h: for *time-spec* grammar, see Table 11, page 90

A.2.1 Aggregate Decomposition

The aggregate decomposition specification is typically defined at the type level, to indicate how an aggregate activity (possibly a wrapper), decomposes into subactivities. Initargs are also passed to subactivities using this tag. It is only necessary to pass initargs for instance slots that are relevant to the recipient. Note also, though, that if an initarg for an instance slot is not passed, its value will be unbound, indicating, in the case of the variables maintained by an activity (i.e., specified using the `<variables>` tag), that they are free. For example, if values for the `resource-origin` and `resource-destination` variables maintained by the `RESOURCE-SOURCING` wrapper activity are not specified by an invoking activity instance, then a resource allocated to this activity may be sourced from anywhere.

The aggregate decomposition specification format is presented in Table 22.

Table 22: Aggregate decomposition specification format

Grammar for *decomposition-spec*			Notes
decomposition-spec	$\rightarrow$	`<decomposition>` [*child-spec*]+ `</decomposition>`	
child-spec	$\rightarrow$	`<child-spec` [`name`=“*child-name*”]`>`	
		type-ref-spec	
		[*initargs-spec*]	**a**
		`</child-spec>`	
type-ref-spec	$\rightarrow$	`<type-ref>` *child-type-name* `</type-ref>`	**b**

a: for *initargs-spec* grammar, see Table 25, page 101

b: *child-type-name* must refer to a predefined activity type

A.2.2 Aggregate Constraints

The aggregate constraint specification, also typically defined at the type level, establishes the sequential relationships among the subactivities of an aggregate activity (again, possibly a wrapper), using the temporal sequencing constraints described in Section A.1.4.1 (**Temporal Sequencing Constraints**). The aggregate constraint specification format is presented in Table 23.

A.2.3 Wrapper Invocation

The wrapper specification is typically defined at the type level, to expand an activity upward (i.e., into superactivities). Similarly to the behavior of an aggregate decomposition, initargs are used to pass slot values up to the wrapper activity instance. The wrapper specification format is presented in Table 24.

Table 23: Aggregate constraint specification format

Grammar for *constraints-spec*			Notes
constraints-spec	→	`<constraints>` [*constraint-spec*]+ `</constraints>`	**a**

a: for *constraint-spec* grammar, see Table 18, page 95

Table 24: Wrapper specification format

Grammar for *wrappers-spec*			Notes
wrappers-spec	→	`<wrappers>` [*wrapper-spec*]+ `</wrappers>`	
wrapper-spec	→	`<wrapper name=`"*wrapper-name*"`>`	**a**
		[*initargs-spec*]	**b**
		`</wrapper>`	

a: *wrapper-name* must refer to a predefined wrapper type

b: for *initargs-spec* grammar, see Table 25, page 101

A.2.4 Initialization Arguments

As has already been mentioned, initialization arguments (i.e., initargs), are used to pass values from one activity to another, whether it be from activity to subactivity or from activity to super-activity. Because COMIREM is built on the Common Lisp Object System (CLOS) [Keene, 1989], the model for initargs reflects the underlying CLOS object model.

Initarg values can be specified in one of three ways:

1. the value of a slot - a *slot-name* is identified, and the value passed with the initarg is the value of that slot belonging to the activity that is passing the initarg

2. an explicit value - the *value* is passed with the initarg

3. a keyword name - a *keyword* is identified, and the value passed with the initarg is the value that was paired with the keyword when the activity that is passing the initarg was instantiated

The *initarg-name* is the name of an initarg that is acceptable to the activity to which the initarg is being passed. The initarg specification format is presented in Table 25.

Table 25: Initialization argument specification format

Grammar for *initargs-spec*			Notes
initargs-spec	→	`<initargs>` [*initarg-spec*]+ `</initargs>`	
initarg-spec	→	`<initarg name=`*initarg-name*`>` *slot-spec* | *value-spec* | *keyword-spec* `</initarg>`	**a**
slot-spec	→	`<slot>` *slot-name* `</slot>`	**b**
value-spec	→	`<value>` *value* `</value>`	**c**
keyword-spec	→	`<keyword>` *keyword* `</keyword>`	**d**

a: *initarg-name* must be an initarg for the targeted activity instance
b: *slot-name* must reference either a slot, variable or input of the current activity instance
c: *value* should be a value appropriate for *initarg-name*
d: *keyword* must be a keyword provided to the current activity instance

A.2.5 Resource Requirements

Resource requirements can be specified at either the type or instance level. The resource-requirement specification tells the scheduler what kinds of resources—and how many of each—must be secured to satisfy an activity. The specification of a capability, model type or resource class automatically includes all of the selection's children (and their children, etc.) as candidate resources with which to satisfy a particular requirement. The optional `count` attribute allows the planner to override the default calculation, performed by the scheduler, that determines how many actual resource instances are required by an activity depending on the nature of its cargo manifest. For example, a resource type with the capability to handle 100 PALLETS, selected for an activity that has a cargo manifest of 400 PALLETS, will require four instances of that resource type to be allocated to the activity.

Currently in COMIREM, each individual resource requirement represents a requirement that must be satisfied. If there are two requirements ($\mathbf{R}_1$ and $\mathbf{R}_2$), then a resource (or resources) satisfying $\mathbf{R}_1$—*and*—a resource (or resources) satisfying $\mathbf{R}_2$ must be secured (i.e., there is an implied *and* operator among multiple requirements).

Note that there are additional semantics involved with the modification of resource requirements, i.e., through the COMIREM GUI. This issue is discussed in Section B.4.1.5.

The resource-requirement specification format is presented in Table 26.

A.2.6 Cargo Manifests

Cargo manifests are generally specified at the instance level. They describe a quantity of cargo that must be accommodated by an activity's required resource(s). Absent a `count` attribute in

Table 26: Resource requirement specification format

Grammar for *requirements-spec*			Notes
requirements-spec	→	`<requirements>` [*requirement-spec*]+ `</requirements>`	
requirement-spec	→	`<requirement` [`count`= "INTEGER"] `>`	**a**
		capabilities-spec | *resource-class-spec*	
		| *model-type-spec* | *resource-spec*	
		`</requirement>`	
capabilities-spec	→	`<capabilities>` [*capability-name*]+ `</capabilities>`	**b**
resource-class-spec	→	`<resource-class>`	
		[*resource-class-name*]+	**c**
		`</resource-class>`	
model-type-spec	→	`<model-type>` [*model-type*]+ `</model-type>`	**d**
resource-spec	→	`<resource>` *resource-name* `</resource>`	**e**

a: the optional *count* attribute overrides the calculation that determines how many resource instances are required for a particular activity

b: *capability-name* names a capability (if it does not yet exist, it will be created)

c: *resource-class-name* must refer to a predefined resource class

d: *model-type* must refer to a predefined model type

e: *resource-name* must refer to an existing resource instance (this is not currently supported)

an activity's resource requirement, the cargo manifest helps drive the calculation to determine how many resource instances are needed to perform an activity. The cargo manifest specification format is presented in Table 27.

A.3 Resource Attributes

The various specification formats for resource attributes are presented in this section. Note that, as in the case of activities, the distinction between type and instance attributes for resources is not strictly defined. Again, attributes specified at the type level will apply to all instances of the type, and attributes specified at the instance level will override type-level definitions. In this section, attributes have been grouped according to how they are typically specified in COMIREM data files.

As should be expected, resource type and instance names must be unique within a scenario. The resource type specification format is presented in Table 28, and the resource instance specification format is presented in Table 29. In the specification of resource instances, note that only `places` can have components (capacitated or not), and that `places` can only be placed at a location specified by a latitude/longitude pair (i.e., they cannot be treated as components).[26]

[26]Note that if a component resource is referenced in a *components-spec*, it must also be defined separately.

Table 27: Cargo manifest specification format

Grammar for *manifest-spec*			Notes
manifest-spec	→	`<manifest>` [*manifest-entry-spec*]+ `</manifest>`	
manifest-entry-spec	→	`<manifest-entry>` *cargo-spec* `</manifest-entry>`	**a**

a: for *cargo-spec* grammar, see Table 17, page 94

Table 28: Resource type specification format

Grammar for *resource-type-spec*			Notes
resource-type-spec	→	`<type>`	
		<*resource-type* `name`=“*new-type-name*”>	
		[*model-type-spec*]	
		[*description-spec*]	
		[*speed-spec*] †	**a**
		[*configurations-spec*]	**b**
		[*capabilities-spec*]	
		</*resource-type*>	
		`</type>`	
resource-type	→	`aircraft` | `landcraft` | `seacraft`	
		| `mobile` | `place` | `component`	
model-type-spec	→	`<model-type>` *model-type* `</model-type>`	**c**
description-spec	→	`<description>` TEXT `</description>`	
capabilities-spec	→	`<capabilities>` [*capability-name*]+ `</capabilities>`	

†: these options are only relevant for mobile resource types

a: for *speed-spec* grammar, see Table 16, page 93

b: for *configurations-spec* grammar, see Table 30, page 105

c: *model-type* must refer to a predefined model type

Table 29: Resource instance specification format

Grammar for *resource-instance-spec*			Notes
resource-instance-spec	→	`<instance>`	
		<*resource-type* `name=`"*resource-name*">	
		type-ref-spec	
		placement-spec	
		[***place-only-options***] †	
		[*capacity-list-spec*]	**a**
		</*resource-type*>	
		`</instance>`	
resource-type	→	`resource` | `place` | `component`	
type-ref-spec	→	`<type-ref>` *type-name* `</type-ref>`	**b**
placement-spec	→	`<placement>`	
		location-spec | *place-name* ‡	**c**, **d**
		`</placement>`	
place-only-options	→	[*components-spec*] [*uncapacitated-components-spec*]	**e**
components-spec	→	`<components>` [*component-name*]+ `</components>`	

†: this option is only relevant for `place` instances
‡: this option is *not* relevant for `place` instances

a: for *capacity-list-spec* grammar, see Table 31, page 106
b: *type-name* must refer to a predefined resource type
c: for *location-spec* grammar, see Table 14, page 92
d: *place-name* should refer to a `place` resource instance
e: for *uncapacitated-components-spec* grammar, see Table 32, page 107

A.3.1 Configurations

Configuration specifications define the capacity of a resource under different setups (i.e., configurations), and are typically defined at the type level. Each configuration option for a resource does two things: (1) it links the resource to a capability, and (2) it provides the information necessary to determine its ability to handle a particular activity based on the activity's cargo manifest. As illustrated in Figure 10, Section 3.2.1.3 (**Sibling Activities**), page 20, and absent a `count` attribute appearing in an activity's resource requirement, the cargo manifest is matched against the relevant resource configuration—designated by the required capability—to determine how many instances of a resource are necessary to perform the activity. If no capability is specified, i.e., the resource requirement instead requests a model type or resource class, then the default resource configuration (named `:default`) is used. If there is no default configuration, a single instance is assumed to be sufficient.

If more than a single *cargo-spec* is defined within a configuration, an operator can be specified to indicate how the multiple *cargo-specs* should be treated when used to determine the required number of resources for a particular cargo manifest. There are two operators: `:and` and `:or`.[27] The use of an operator facilitates a richer representation of configurations, by allowing for the case where a resource can be set up in such a way as to accommodate different combinations of cargo at the same time, instead of only one (i.e., configuration $\mathbf{C}_1$ can handle a maximum of 6 PALLETS, while configuration $\mathbf{C}_2$ can handle a maximum of 20 PAX *and* 2 vehicles. If no operator is specified, the default (i.e., `:or`) is used.

The configuration specification format is presented in Table 30.

Table 30: Resource configuration specification format

Grammar for *configurations-spec*			Notes
configurations-spec	→	`<configurations>`	
		[*configuration-spec*]+	
		`</configurations>`	
configuration-spec	→	`<configuration` *name-spec* [*operator-spec*] `>`	
		[*cargo-spec*]+	**a**
		`</configuration>`	
name-spec	→	`name=(`"`:default`" | "*capability-name*"`)`	**b**
operator-spec	→	`op=(`"`or`" | "`and`"`)`	**c**

a: for *cargo-spec* grammar, see Table 17, page 94

b: *capability-name* names a capability (if it does not yet exist, it will be created)

c: the `:and` operator is not currently supported

[27] Note that the `:and` operator is not currently supported by COMIREM.

A.3.2 Capacity

The capacity specification describes the availability of a resource over time, and is generally defined at the instance level. Capacity is specified as a sequence of (typically) contiguous (but always non-overlapping) time intervals indicating how many units of a resource are available for allocation by the scheduler at any point in time. COMIREM currently supports the definition of *unitary-capacity, reusable* resources, i.e., they provide one unit of capacity and become available for reuse after completing any service to which they have been allocated. An activity may, however, require more than a single unit of capacity, in which case multiple resource instances will be required.

Each capacity interval includes a `start-time` and `end-time` tag, the value of which may be a valid `time-spec`, or one of the special keywords: `:negative-infinity` or `infinity`, which represent unbounded negative and positive times, respectively.[28] The capacity specification format is presented in Table 31.

Table 31: Capacity specification format

Grammar for *capacity-spec*			Notes
capacity-spec	$\rightarrow$	`<capacity-list>`	
		[*capacity-interval-spec*]+	
		`</capacity-list>`	
capacity-interval-spec	$\rightarrow$	`<capacity-interval>`	
		start-spec *end-spec* *units-spec*	
		`</capacity-interval>`	
start-spec	$\rightarrow$	`<start-time>`	
		(*time-spec* | "`:negative-infinity`")	**a**
		`</start-time>`	
end-spec	$\rightarrow$	`<end-time>`	
		(*time-spec* | "`:infinity`")	**a**
		`</end-time>`	
units-spec	$\rightarrow$	`<units>` INTEGER `</units>`	**b**

a: for *time-spec* grammar, see Table 11, page 90

b: at the present time, the `units` integer value must be 1 (one).

[28]Note that all time intervals must be positive (i.e., `:negative-infinity` $<$ *start-time* $\leq$ *end-time* $<$ `:infinity`).

A.3.3 Uncapacitated Place Components

Uncapacitated place components are generally defined at the instance level, and only for `place` resources. Any `place` resource may provide a bucket of uncapacitated resource capacity, specified in terms of capabilities, model types and resource classes.

The uncapacitated place component specification format is presented in Table 32.

Table 32: Uncapacitated place components specification format

Grammar for *uncapacitated-components-spec*			Notes
uncapacitated-components-spec	$\rightarrow$	`<uncapacitated-components>`	
		capabilities-spec	
		| *resource-class-spec*	
		| *model-type-spec*	
		`</uncapacitated-components>`	
capabilities-spec	$\rightarrow$	`<capabilities>`	
		[*capability-name*]+	**a**
		`</capabilities>`	
resource-class-spec	$\rightarrow$	`<resource-class>`	
		[*resource-class-name*]+	**b**
		`</resource-class>`	
model-type-spec	$\rightarrow$	`<model-type>`	
		[*model-type*]+	**c**
		`</model-type>`	

a: *capability-name* names a capability (if it does not yet exist, it will be created)
b: *resource-class-name* must refer to a predefined resource class
c: *model-type* must refer to a predefined model type

A.4 Capabilities

Capability specifications appear in the *situation*-`capabilities.xml` file, and define capabilities that are linked to resource classes within a particular situation for reference by both the activity types in that situation and the activity instances in a particular scenario.

The `type-ref` value of the *type-ref-spec* entry may be either `move` or `event`. The purpose of this entry is to inform the system about the nature of the capability being defined (i.e., movement capabilities are for `move`s, while stationary capabilities are for `event`s).

The capabilities specification format is presented in Table 33.

Table 33: Capability specification format

Grammar for *capability-spec*			Notes
capability-spec	$\rightarrow$	`<capability` *name-spec*`>`	
		capable-types-spec *type-ref-spec*	
		`</capability>`	
name-spec	$\rightarrow$	`name`=“SYMBOL”	
capable-types-spec	$\rightarrow$	`<capable-types>`	
		[*resource-class-name*]+	**a**
		`</capable-types>`	
type-ref-spec	$\rightarrow$	`<type-ref>` (`move` | `event`) `</type-ref>`	

a: *resource-class-name* must refer to a predefined resource class

A.5 Plan Configuration

The plan-configuration specification appears at the start of the *scenario*-`plan.xml` file, ahead of the scenario-specific `place`, `activity` and `constraint` instances. It provides mostly temporal information about the plan to facilitate its processing by COMIREM.

The plan-configuration specification format is presented in Table 34.

Table 34: Plan-configuration specification format

Grammar for *plan-configuration-spec*			Notes
plan-configuration-spec	→	`<configuration>`	
		[*display-name-spec*]	
		[*calendar-zero-spec*]	
		[*calendar-zero-date-spec*]	
		[*ref-hours-spec*]	**a**
		[*plan-horizon-spec*]	
		[*time-granularity-spec*]	
		`</configuration>`	
display-name-spec	→	`<display-name>` TEXT `</display-name>`	
calendar-zero-spec	→	`<calendar-zero>` INTEGER `</calendar-zero>`	
calendar-zero-date-spec	→	`<calendar-zero-date>`	
		time-spec	**b**
		`</calendar-zero-date>`	
plan-horizon-spec	→	`<plan-horizon>` *interval-spec* `</plan-horizon>`	**c**
time-granularity-spec	→	`<time-granularity>`	
		time-granularity	
		`</time-granularity>`	
time-granularity	→	`seconds` | `minutes` | `hours` | `days`	

a: for *ref-hours-spec* grammar, see Table 19, page 96

b: for *time-spec* grammar, see Table 11, page 90

c: for *interval-spec* grammar, see Table 12, page 91

B COMIREM User Manual

The COMIREM system consists of two components: a server and a client. The server is a self-contained executable program running a Common Lisp-based, HTTP-compliant web server. The client is an HTML file that contains an embedded Shockwave application.

B.1 Installation

To install COMIREM on a Microsoft Windows-based computer, run the installation program: `COMIREMInstaller.exe`. This will install the COMIREM system onto your computer (by default, all files will be placed in the `C:\Program Files\Comirem` directory) and establish all necessary environment variables (namely, `OZONE_HOME`).

On an Apple Macintosh or Sun Solaris platform, COMIREM is launched from within a Franz Allegro Common Lisp session, with the necessary Ozone and COMIREM code loaded and the `OZONE_HOME` environment variable properly set.[29]

B.2 Starting COMIREM

To start COMIREM in a Windows environment, execute `Comirem.exe`. In a Macintosh or Solaris environment, execute the `start-comirem` function from inside a properly initialized Common Lisp session. This will start the server, which is then ready to accept connections on port 8000.[30]

To run the client, open `COMIREM.htm` in any Internet browser equipped with an up-to-date Shockwave plug-in.[31]

B.3 Running COMIREM

We begin with a description of the buttons at the top of the **Activity Network Display** (the initial COMIREM display). These buttons, shown in Figure 57, provide basic functionality for operating the COMIREM client.

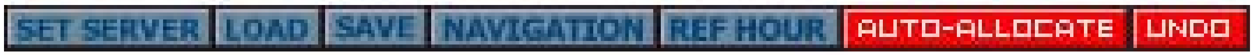


Figure 57: COMIREM GUI navigational buttons

Note that the $\boxed{\text{NAVIGATION}}$ and $\boxed{\text{REF HOUR}}$ buttons are discussed within the context of the **Activity Network Display**, in Sections B.4.1.1 and B.4.1.6, respectively.

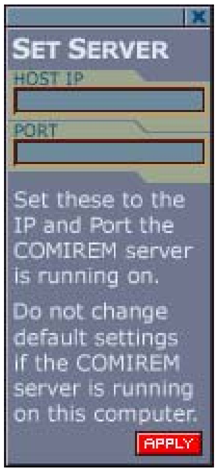
[29] In the absence of an executable file, the source code can be loaded into an Ozone-initialized Lisp environment using the Ozone-provided `load-comirem` function.

[30] Note that when starting from within a Common Lisp session, the function `start-comirem` accepts a `:PORT` keyword argument for specifying the desired port number (which defaults to 8000).

[31] Note that Netscape version 6.0 does not support Shockwave.

B.3.1 SET SERVER Button

The first step is to establish communication between the COMIREM client and the server. When started, the *Set Server* pop-up window will appear, as shown in Figure 58.[32] The default settings assume that both the server and client are running on the same computer. If this is the case, leave the default settings unchanged and press APPLY. If not, set the HOST IP and PORT fields to the IP address and port where the COMIREM server is running and then press APPLY.



Set these to the IP and Port the COMIREM server is running on.

Do not change default settings if the COMIREM server is running on this computer.

Figure 58: *Set Server* pop-up window

B.3.2 LOAD Button

The next step is to load a scenario and any number of plans. Pressing the LOAD button will open the *Load* pop-up window shown in Figure 59. One scenario and any number of plans can be loaded at any time. Scenarios define the available resource types, capabilities and resource bed-down information (i.e., available resource and place instances). Plans define the activities, constraints and plan-specific places that comprise the activity network.[33]

When the *Load* pop-up window appears for the first time, only scenarios are listed. Once a scenario is selected, a list of the plans appropriate for the selected scenario appears in the list on the right. The scenario and plan(s) that are already loaded are highlighted in red. Any number of plans may be selected. Selected scenario and plan(s) are highlighted in red. To unselect, click on any highlighted scenario or plan.

The *Load* pop-up window provides four menu options for loading data files:

- LOAD FROM SCRATCH

  Loads the selected scenario and plan(s), disposing of any existing context

- ADJUST PLANS

  Loads the selected plan(s) into the current scenario and removes any deselected plans

[32] This pop-up window can be accessed at any time by pressing the SET SERVER button.
[33] All scenario and plan files are located in the `OZONE_HOME/data/comirem` directory.

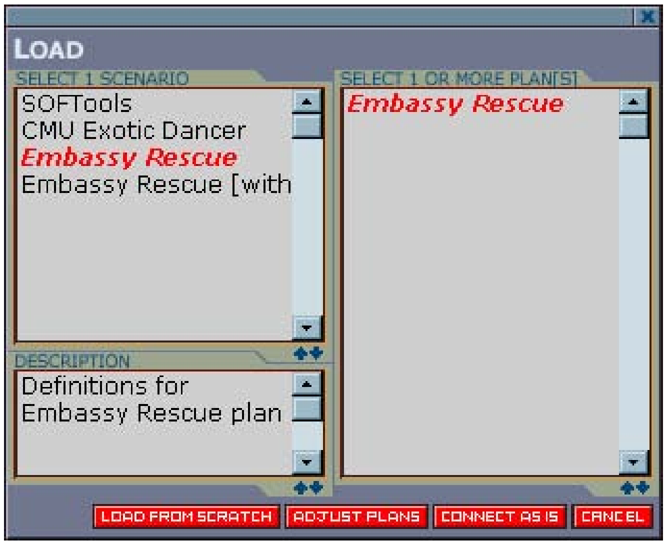


Figure 59: *Load* pop-up window

- CONNECT AS IS

  Connects to the server and retrieves the activity network in its current state (this is useful if the user has closed and then reopened the browser, intending to continue from where they left off)

- CANCEL

  Exits the *Load* pop-up window without loading any scenario or plan(s)

B.3.3 SAVE Button

Saving a schedule is not actively supported through the COMIREM GUI at this time.

B.3.4 AUTO-ALLOCATE Button

Pressing the AUTO-ALLOCATE button causes the server to attempt to schedule all currently unscheduled activities in the currently loaded plan. The server will not schedule any activities with conflicts (see Section B.4.1).

B.3.5 UNDO Button

As the name implies, the UNDO button undoes the last action taken by the user and discards any consequences of that action.

B.4 Client Components

The COMIREM GUI client is comprised of four principle displays: the **Activity Network Display**, the **Resource & Taskforce Manager**, the **Place Manager**, and the **Magboard**. Access to each display is obtained by clicking on its name in the client component selector, which is included at the bottom of each display and shown in Figure 60.

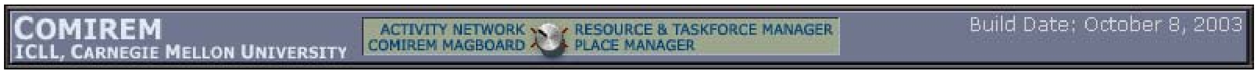

COMIREM
ICLL, CARNEGIE MELLON UNIVERSITY
ACTIVITY NETWORK
COMIREM MAGBOARD
RESOURCE & TASKFORCE MANAGER
PLACE MANAGER
Build Date: October 8, 2003

Figure 60: GUI client component selector

In the remainder of this section, we describe the features of each of these four displays.

B.4.1 Activity Network Display

Once a scenario and plan(s) are loaded, the **Activity Network Display** is populated with the collective activity network (i.e., the network comprising all loaded plans) and is automatically displayed. The **Activity Network Display** can be presented in two different formats: the (default) *Gantt View* described in Section B.4.1.2, and the *Vector View* described in Section B.4.1.3. All components of the activity network are colored according to their status, as described in Section B.4.1.4.

B.4.1.1 NAVIGATION Button

Alternation between the two views of the **Activity Network Display** is achieved using the *Navigation* pop-up window shown in Figure 61, which is reached by clicking on the top-level NAVIGATION button. The magnifying glass buttons zoom in and out on the activity network. To zoom to the scale at which the entire temporal horizon of the activity network is visible, press the SHOW ALL TIMES button. The result of this action only ensures that the entire activity network fits horizontally in the display—not vertically (i.e., it may still be necessary to scroll up and down to see activities outside of the screen).

The *Viewport* displays a miniature version of the entire activity network. Clicking on a point within it re-centers the **Activity Network Display** (either the *Gantt* or *Vector View*) on that point. The *Viewport* utilizes the same color coding as the **Activity Network Display**. The toggle switch for alternating between the *Gantt View* and *Vector View* is located below the *Viewport*. Below that, depending on the current view, is a final option for modifying the display. In the *Vector View*, the user can choose between displaying the activity network using earliest finish times (EFTs) or latest finish times (LFTs, the default). In the *Gantt View*, the user can choose between an enhanced and a standard temporal display (this is discussed in Section B.4.1.2).

B.4.1.2 Gantt View

In the *Gantt View*, shown in Figure 62, each activity and thread in the activity network is represented by a single horizontal bar. Places are not shown. Threads are represented by purple bars and only their start and finish times are displayed. The constituent activities of each thread are displayed underneath it, sorted by start time, and the thread can be collapsed or expanded by clicking on its name. Within each activity bar, the lighter color represents the overall time window within which an activity must be completed, and the darker color represents the duration of the activity. As mentioned above, toggling the RICH TIMES button in the *Navigation* pop-up

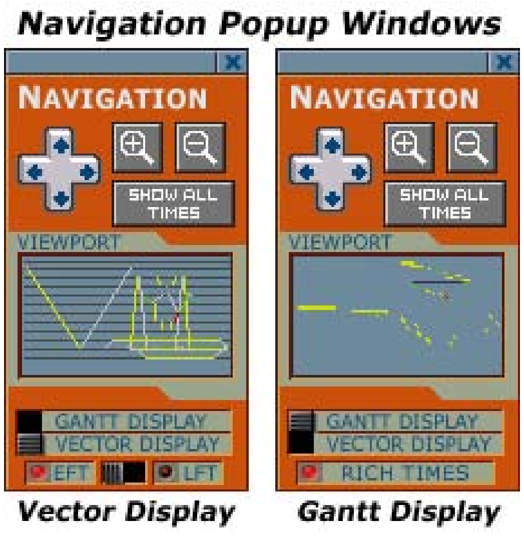


Figure 61: *Vector View* and *Gantt View* display panes for the *Navigation* pop-up window

window switches between an enhanced temporal display and the standard (default) format. In the enhanced format, each activity bar shows both its earliest and latest start times (also called "no earlier than" or NET time, and "no later than" or NLT time, respectively), earliest and latest end times and its minimum and maximum duration, as explained in Figure 63.

B.4.1.3 Vector View

The *Vector View* of the **Activity Network Display** is shown in Figure 64. It depicts a basic "plan-oriented" visualization of an activity network and has been adapted from other current SOF planning tools. Horizontal bars are used to indicate an activity (or more generally, a sequence of activities) taking place at a particular location over time, and diagonal bars are used to indicate movement (or transport) between locations. Threads are currently not displayed in the *Vector View*.

B.4.1.4 Color Legend

The color legend for the **Activity Network Display** is presented in Figure 65.

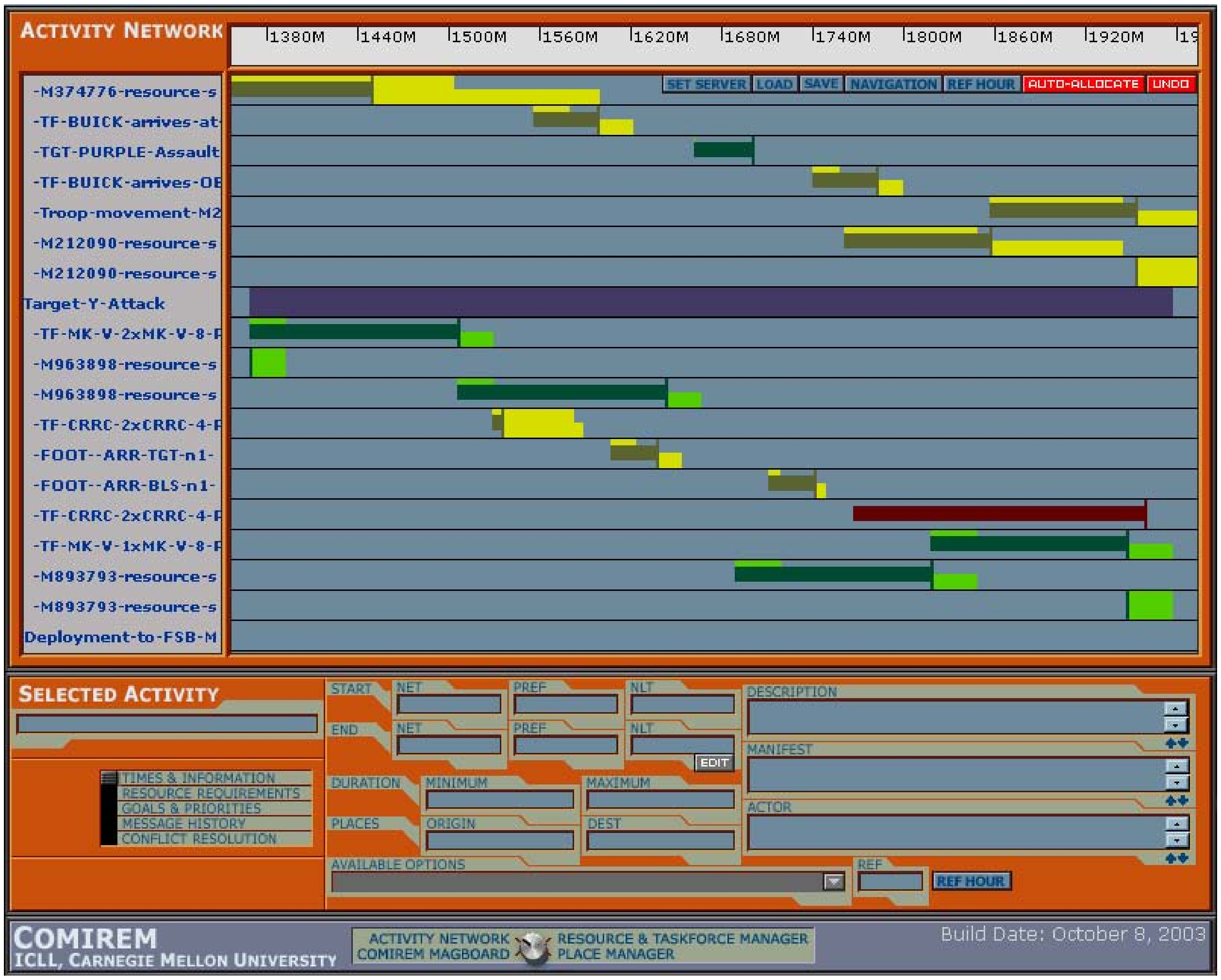


Figure 62: **Activity Network Display** (*Gantt View*)

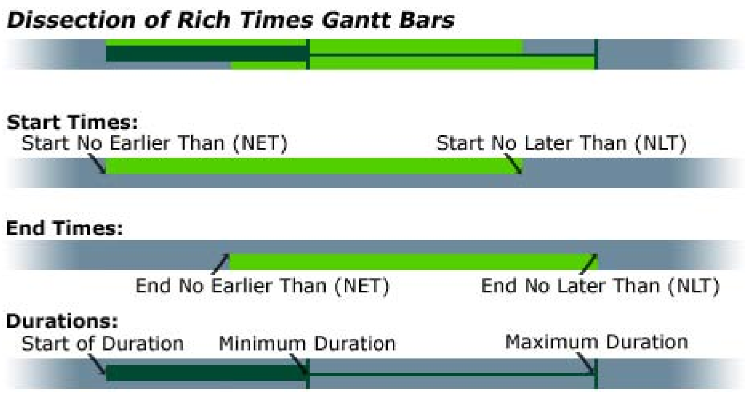

Dissection of Rich Times Gantt Bars
Start Times:
Start No Earlier Than (NET)
Start No Later Than (NLT)
End Times:
End No Earlier Than (NET)
End No Later Than (NLT)
Durations:
Start of Duration
Minimum Duration
Maximum Duration

Figure 63: Details of the enhanced temporal display in the *Gantt View*

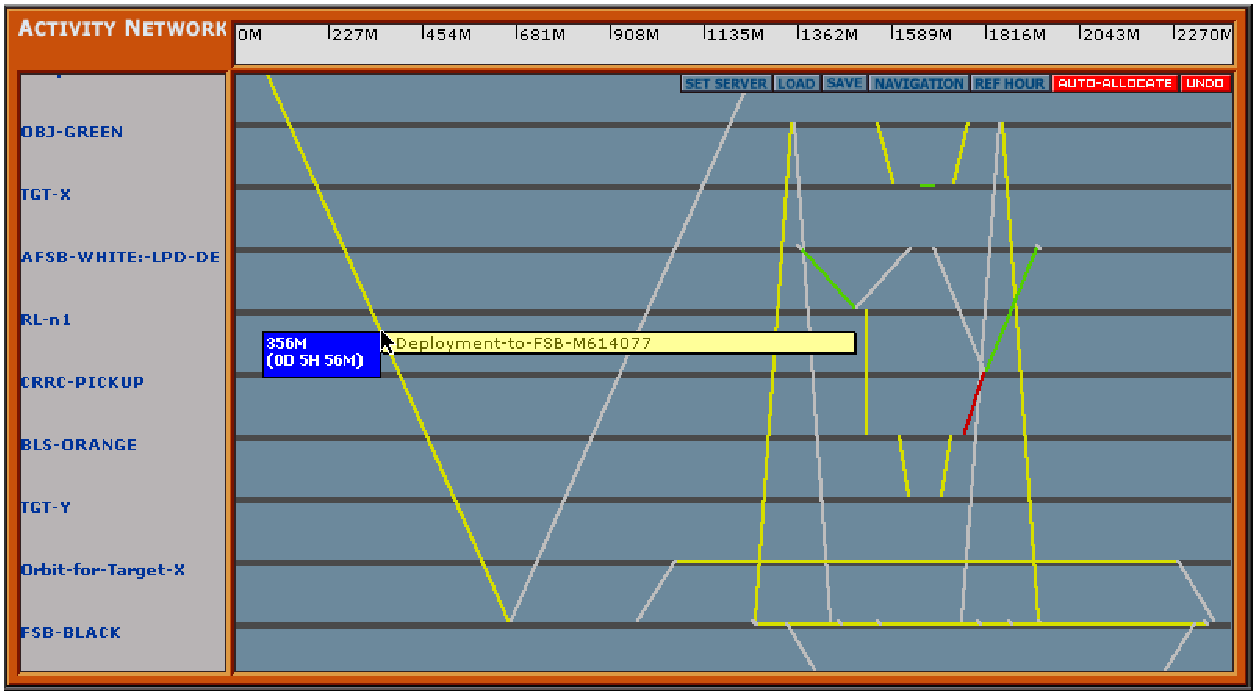
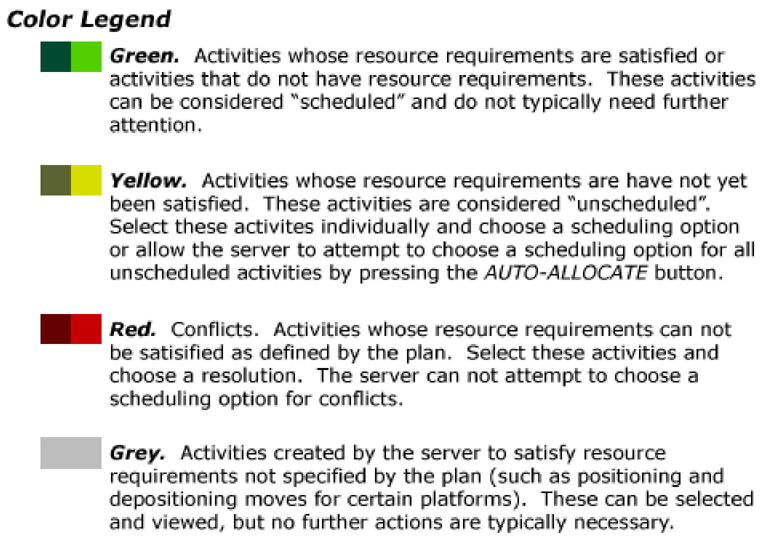


Figure 64: **Activity Network Display** (*Vector View*)



Color Legend

🟩 ***Green.*** Activities whose resource requirements are satisfied or activities that do not have resource requirements. These activities can be considered "scheduled" and do not typically need further attention.

🟨 ***Yellow.*** Activities whose resource requirements are have not yet been satisfied. These activities are considered "unscheduled". Select these activites individually and choose a scheduling option or allow the server to attempt to choose a scheduling option for all unscheduled activities by pressing the *AUTO-ALLOCATE* button.

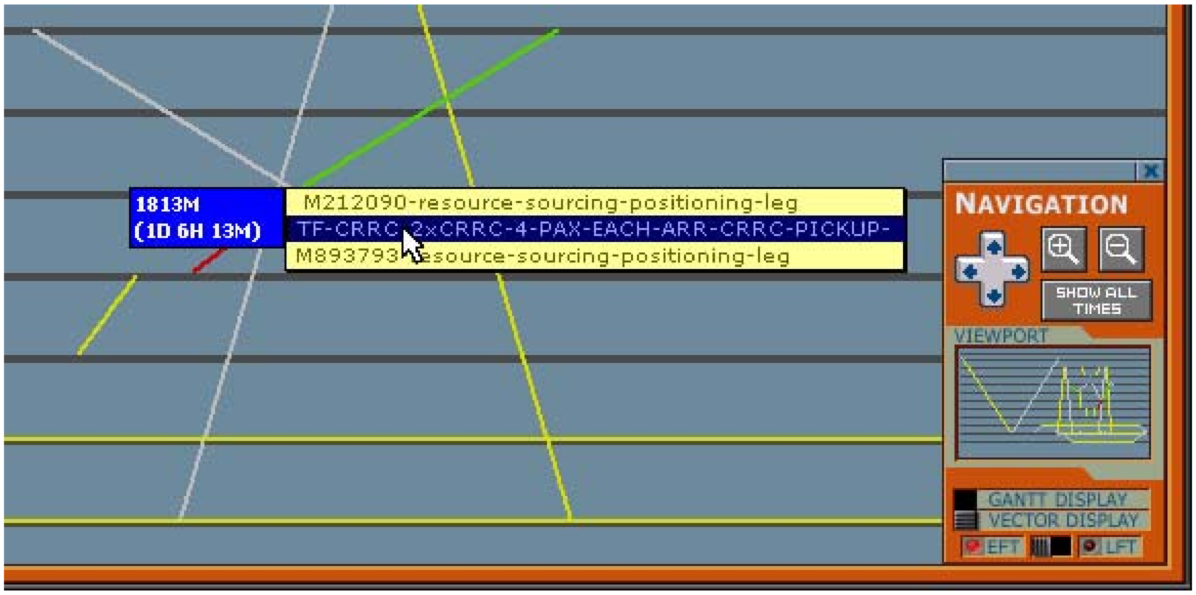
🟥 ***Red.*** Conflicts. Activities whose resource requirements can not be satisified as defined by the plan. Select these activities and choose a resolution. The server can not attempt to choose a scheduling option for conflicts.

⬜ ***Grey.*** Activities created by the server to satisfy resource requirements not specified by the plan (such as positioning and depositioning moves for certain platforms). These can be selected and viewed, but no further actions are typically necessary.

Figure 65: **Activity Network Display** color legend

B.4.1.5 Selected Activity Form

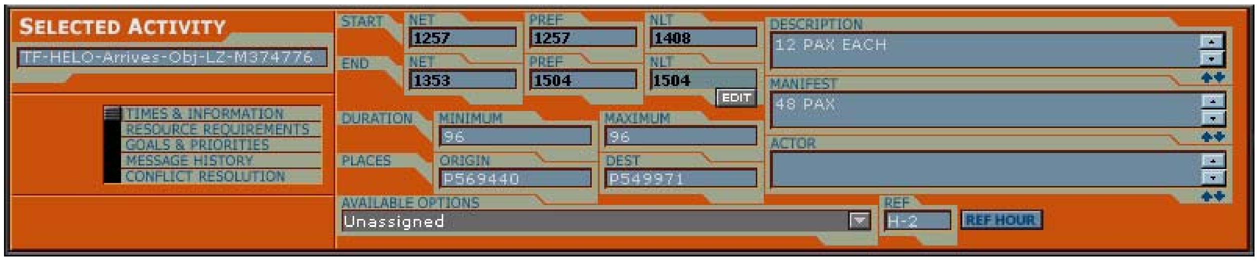
An activity (or thread) can be selected in the *Gantt View* by clicking on its corresponding bar. In the *Vector View*, an activity is selected by positioning the cursor over the desired activity (or multiple collocated activities), at which point a yellow pop-up menu will appear to the right of the cursor listing all of the activities within close range. The desired activity is selected by first clicking and holding down the mouse button to secure the pop-up menu, positioning the cursor over the name of the desired activity, and finally releasing the mouse button. Figure 66 depicts the middle step in this process.



1813M
(1D 6H 13M)
M212090-resource-sourcing-positioning-leg
TF-CRRC 2xCRRC-4-PAX-EACH-ARR-CRRC-PICKUP-
M893793 resource-sourcing-positioning-leg
NAVIGATION
SHOW ALL TIMES
VIEWPORT
GANTT DISPLAY
VECTOR DISPLAY
EFT
LFT

Figure 66: Selecting an activity in the *Vector View* of the **Activity Network Display**

When an activity or thread is selected, the *Selected Activity* form at the bottom of the **Activity Network Display** is automatically populated with the data for the selected object, as shown in Figure 67.



SELECTED ACTIVITY
TF-HELO-Arrives-Obj-LZ-M374776

TIMES & INFORMATION
RESOURCE REQUIREMENTS
GOALS & PRIORITIES
MESSAGE HISTORY
CONFLICT RESOLUTION

START	NET 1257	PREF 1257	NLT 1408
END	NET 1353	PREF 1504	NLT 1504
DURATION	MINIMUM 96	MAXIMUM 96	
PLACES	ORIGIN P569440	DEST P549971	

EDIT

DESCRIPTION: 12 PAX EACH
MANIFEST: 48 PAX
ACTOR:
AVAILABLE OPTIONS: Unassigned
REF: H-2
REF HOUR

Figure 67: Information about a selected activity in the TIMES & INFORMATION pane of the *Selected Activity* form in the **Activity Network Display**

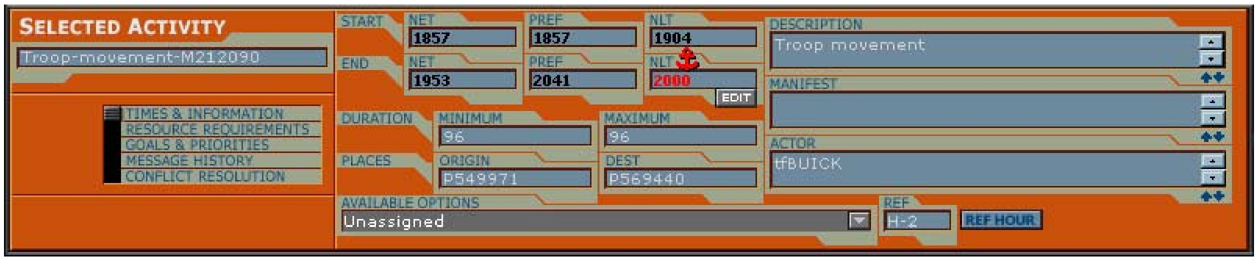
The *Selected Activity Form* consists of five panes to convey information to the user and facilitate interaction. Each of these panes is accessible using the thumb-toggle interface on the left-hand side of the form:

- TIMES & INFORMATION

  As shown in Figure 67, this pane conveys all temporal and general information for an activity or thread. For activities and threads, this includes the start and end times and a description of the object. For activities, it also includes duration, anchor and ref-hour constraints, positioning, manifest and actor (i.e., an assigned taskforce) information, and a pull-down menu of feasible scheduling options. For threads, it also includes a pull-down menu of feasible taskforce assignments.

  An activity can be scheduled by selecting a scheduling option from the AVAILABLE OPTIONS pull-down menu. A taskforce can be assigned to a thread using the same menu.

  It is possible to override the latest finish time (LFT), or END NLT (i.e., end no later than) time, of a selected activity by establishing an *anchor* on that time point. An NLT anchor further constrains the latest finish time dictated by the underlying temporal network by introducing a new hard upper-bound constraint fixed at a certain time. To set an anchor, simply edit the time in the END NLT field and press the EDIT button. The **Activity Network Display**, *Selected Activity* form, and most importantly, the selected activity's feasible scheduling options, will all be updated accordingly. When an anchor is set, a small red anchor icon appears above the END NLT time, as depicted in Figure 68. An anchor can be removed by clicking on the anchor icon.[34]



Troop-movement-M212090

	NET	PREF	NLT
START	1857	1857	1904
END	1953	2041	2000

	MINIMUM	MAXIMUM
DURATION	96	96

	ORIGIN	DEST
PLACES	P549971	P569440

DESCRIPTION: Troop movement
MANIFEST:
ACTOR: tfBUICK
AVAILABLE OPTIONS: Unassigned
REF: H-2

Figure 68: Setting an NLT anchor on a selected activity in the TIMES & INFORMATION pane of the *Selected Activity* form in the **Activity Network Display**

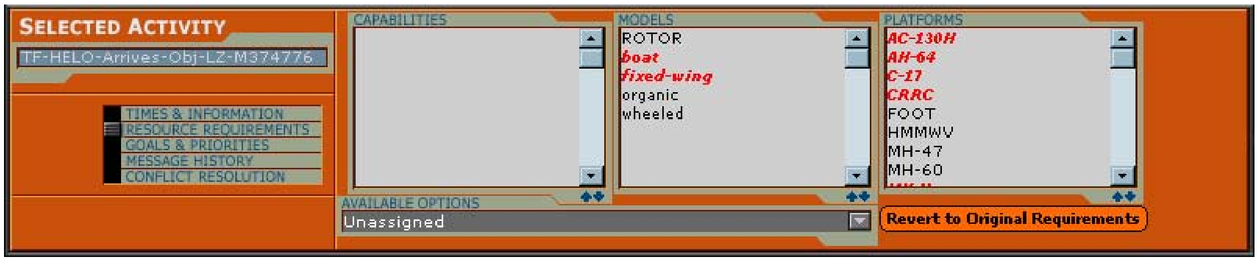
  In the *Vector View* of the **Activity Network Display**, anchors are represented by a purple line/icon extending up from the end of the activity to the anchor time. In the *Gantt View*, the length of the bar for the activity is adjusted to account for an anchor (i.e., there is no special icon).

- RESOURCE REQUIREMENTS

  The RESOURCE REQUIREMENTS pane, depicted in Figure 69, provides the means for interactively modifying the resource requirements for an activity. The feasible scheduling options

[34]The setting of EST, LST and EFT anchors is not currently supported in the COMIREM GUI, although these anchor types are supported by COMIREM itself.

in the AVAILABLE OPTIONS pull-down menu can be changed by relaxing or tightening the requirements in this pane.



TF-HELO-Arrives-Obj-LZ-M374776

CAPABILITIES	MODELS	PLATFORMS
	ROTOR	*AC-130H*
	boat	*AH-64*
	fixed-wing	*C-17*
	organic	*CRRC*
	wheeled	FOOT
		HMMWV
		MH-47
		MH-60

AVAILABLE OPTIONS: Unassigned

Figure 69: Resource requirements for a selected activity in the RESOURCE REQUIREMENTS pane of the *Selected Activity* form in the **Activity Network Display**

The feasible scheduling options for an activity are initially constrained by the set of allowable resource classes indicated by the highlighted platforms in this pane. The set of platforms (i.e., resource classes) is determined by selecting and deselecting capabilities, models and platforms. (Selected items are highlighted in red italics.) Changes in one column may affect the selections in other columns. The semantics of this process are as follows. Referring back to Figure 16, Section 3.2.2.1 (**The Resource Requirement Hierarchy**), page 29, the hierarchy puts capabilities at the top and resource classes (platforms) at the bottom. Whenever a capability, model or platform is selected (or deselected), its children are selected (deselected) as well. The highest level at which any item is specified controls the selection process. If two items at the same level are selected, the *union* of their descendents (i.e., platforms and possibly models) is selected. If a selection is made at a higher level, *only its descendents* will remain selected. If an item is deselected, its descendents are deselected unless they remain feasible owing to the reapplication of the selection semantics.

The REVERT TO ORIGINAL REQUIREMENTS button can be used to restore the original resource-requirement specifications at any time. Similarly, an activity can be scheduled by selecting a scheduling option from the AVAILABLE OPTIONS pull-down menu.

- GOALS & PRIORITIES

  This pane is intended to allow the user to specify goals and priorities for an activity to affect its treatment by the scheduler. *This feature is not currently supported.*

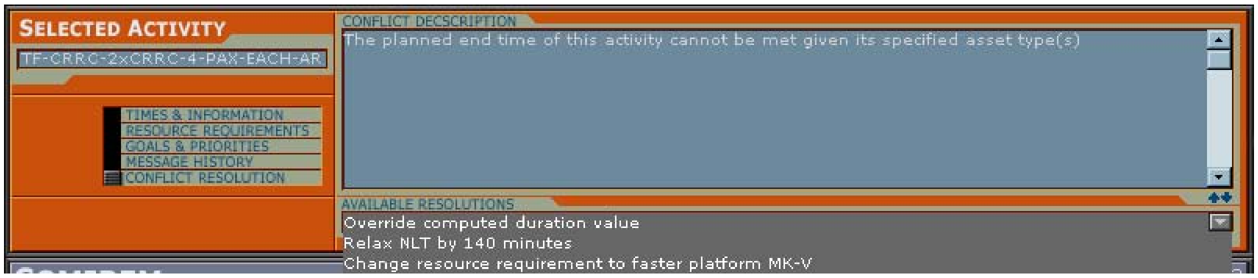
- MESSAGE HISTORY

  This pane is intended to provide a history of all decisions and scheduling actions involving an activity. *This feature is not currently supported.*

- CONFLICT RESOLUTION

  It is possible for the server to determine that an activity cannot be completed within the time constraints defined by the plan given the activity's specified resource requirements.

These activities are considered to be *in conflict* and are displayed in red in the **Activity Network Display**. For a conflicted activity, the CONFLICT RESOLUTION pane, shown in Figure 70, provides a description for the conflict and the means available for its resolution.



The planned end time of this activity cannot be met given its specified asset type(s)

Override computed duration value
Relax NLT by 140 minutes
Change resource requirement to faster platform MK-V

Figure 70: Conflict-resolution options for a selected (conflicted) activity in the CONFLICT RESOLUTION pane of the *Selected Activity* form in the **Activity Network Display**

The AVAILABLE RESOLUTIONS pull-down menu provides a list of available resolution options for the conflicted activity. When a resolution from this list is selected, the conflict is resolved and feasible scheduling options can again be determined for the activity.

B.4.1.6 REF HOUR Button

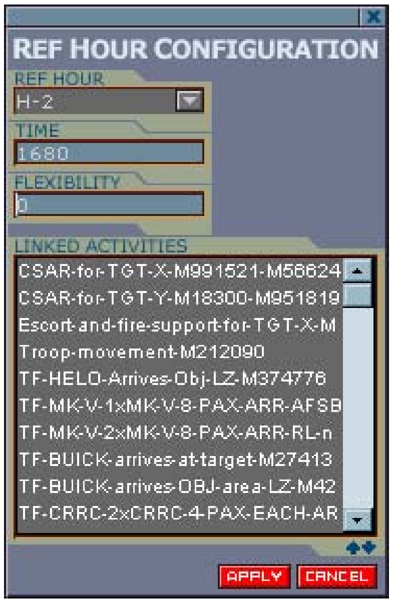
As described in Section 3.2.3.2 (**Reference Hour Constraints**), activities can be linked to plan-specific ref-hours. A ref-hour is centered on a specific point in time, and all activities linked to it (via their start/end times) share its temporal flexibility. A ref-hour defines *preferred* times for an activity, lying within its [EST..LST] and [EFT..LFT] ranges. Referring back to Figure 68 (page 118), note that the link to ref-hour `H-2` results in a preferred start time of 1857 and a preferred end time of 2041 for the selected activity. Not all activities are linked to ref-hours, and not all plans have ref-hours.

Pressing the REF HOUR button opens the *Ref Hour Configuration* pop-up window depicted in Figure 71. From this window, a ref-hour can be selected to access its time, temporal flexibility, and all activities linked to it. The time and flexibility of the selected ref-hour can be modified in the TIME and FLEXIBILITY fields, respectively. When the APPLY button is pressed, all changes are applied to, and propagated throughout, the activity network.

B.4.1.7 Execution Results

COMIREM processes execution results by accepting actual execution times (also referred to as *call-times*) in its situation files and interpreting them as hard temporal constraints (i.e., immovable anchors) on the start/end times of both executed and in-process activities. The latest calltime determines the value of the *current-time* clock in COMIREM. When calltimes are specified for a given situation, the plan is considered to be in progress, and the latest calltime is marked in the **Activity Network Display** with a vertical red line, as shown in Figure 72. The portion of the screen to the left of this line is shaded in gray, indicating that all activities executing in this region are considered to have completed or be in-process.[35]

[35]Note that this implements a rather simple model for processing execution results: if a start calltime has not


REF HOUR CONFIGURATION

REF HOUR
H-2

TIME
1680

FLEXIBILITY
0

LINKED ACTIVITIES
CSAR-for-TGT-X-M991521-M56624
CSAR-for-TGT-Y-M18300-M951819
Escort-and-fire-support-for-TGT-X-M
Troop-movement-M212090
TF-HELO-Arrives-Obj-LZ-M374776
TF-MK-V-1xMK-V-8-PAX-ARR-AFSB
TF-MK-V-2xMK-V-8-PAX-ARR-RL-n
TF-BUICK-arrives-at-target-M27413
TF-BUICK-arrives-OBJ-area-LZ-M42
TF-CRRC-2xCRRC-4-PAX-EACH-AR

Figure 71: *Ref Hour Configuration* pop-up window

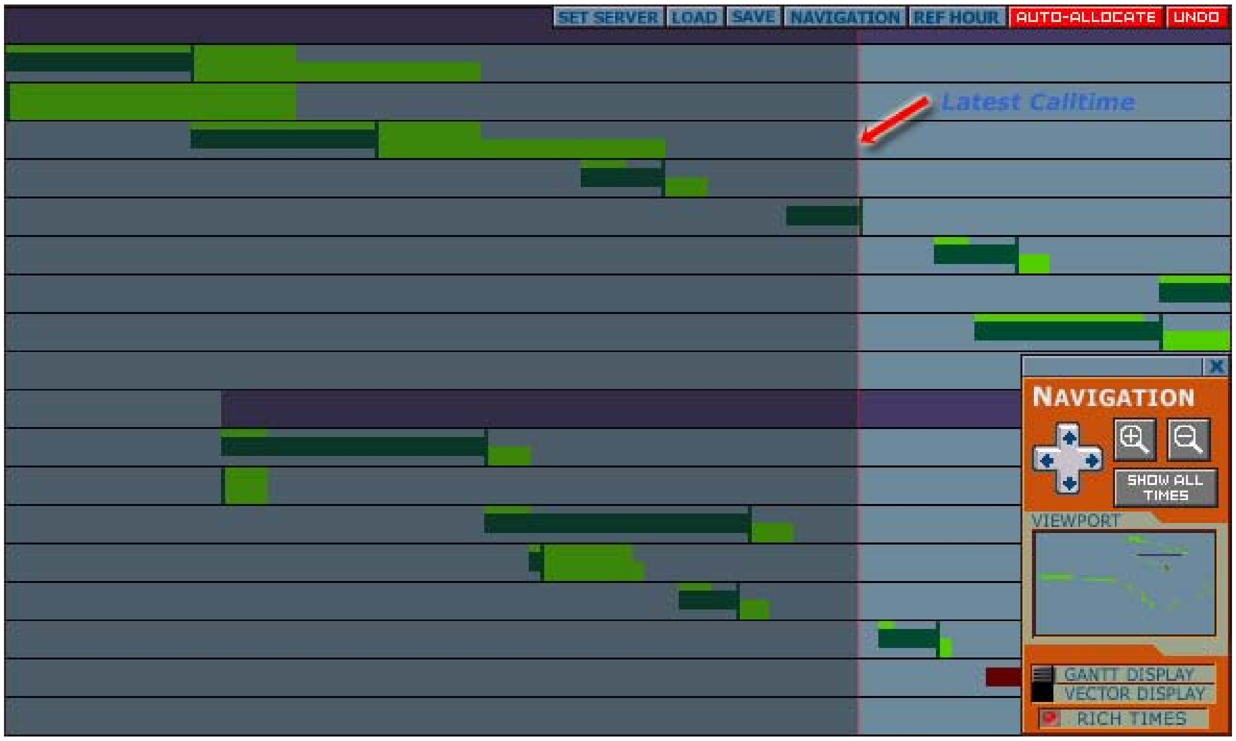


Figure 72: The latest calltime reflected in the *Gantt View* of the **Activity Network Display**

B.4.2 Resource & Taskforce Manager

The **Resource & Taskforce Manager** provides access to the available resources and taskforces within a given scenario, to facilitate their creation, viewing and modification. The **Resource & Taskforce Manager** provides two separate forms: one for resource-capacity and attribute manipulation (the *Resource Manager*) and one for taskforce manipulation (the *Taskforce Manager*). Note that currently, the scope of any modifications made using these forms is limited to the current planning/scheduling session, that is, any changes made here cannot yet be saved to a file.

B.4.2.1 Resource Manager

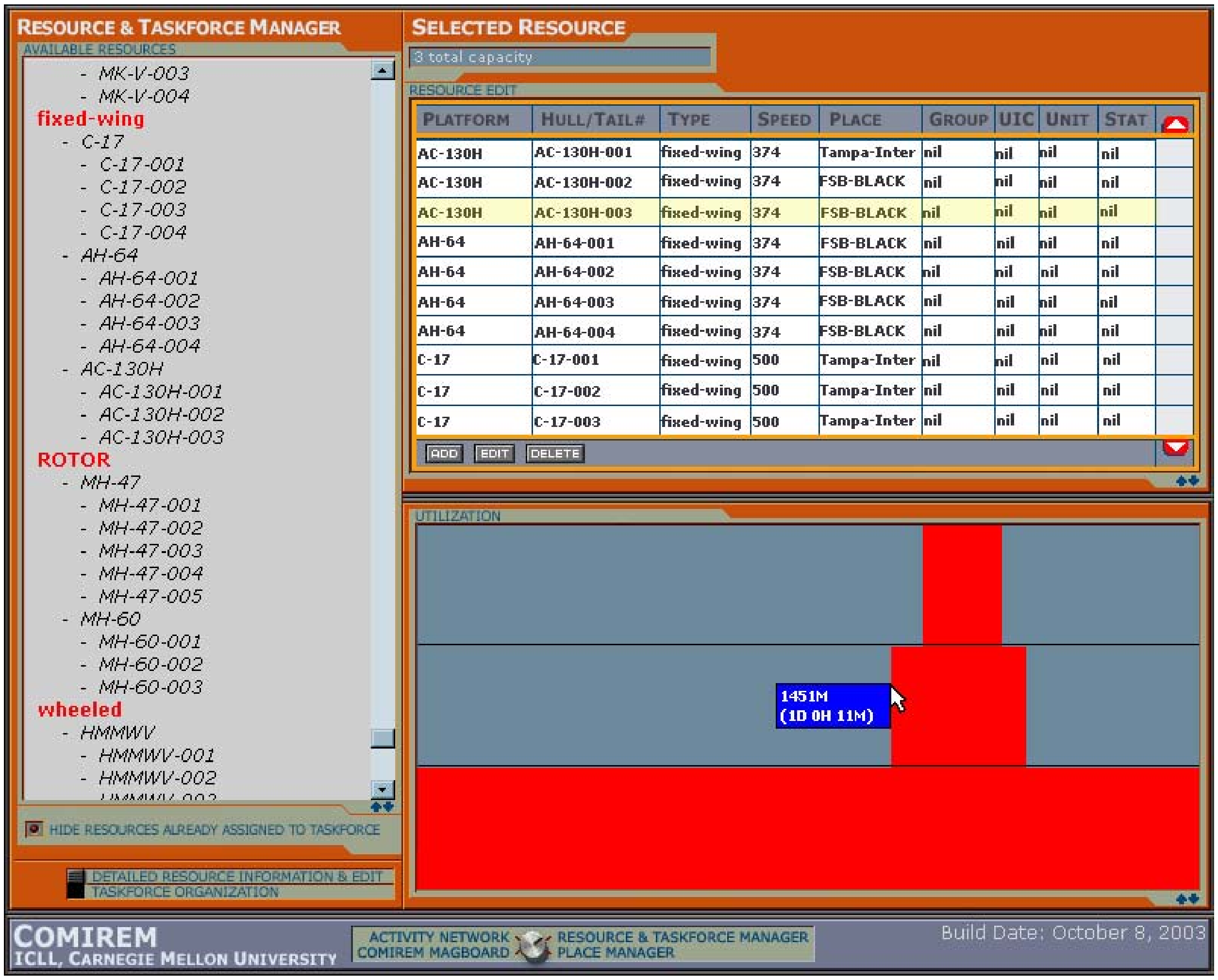
The *Resource Manager*, depicted in Figure 73, provides a form for viewing resource capacity, editing resource attributes, and creating and deleting resource instances. On the left side of the form, the AVAILABLE RESOURCES column lists all known resources, sorted and indexed by model and platform. Clicking on any model, platform, or resource causes its capacity line to be displayed in the UTILIZATION window on the lower right side of the form. The UTILIZATION window displays the entire time horizon for the loaded plan(s), and includes one row for each instance corresponding to the item (e.g., model, platform, resource) selected. The red blocks represent used capacity (i.e., indicating existing reservations), while the gray blocks represent available capacity.

In the upper right side of the form, the RESOURCE EDIT table facilitates the creation, editing and deletion of individual resource instances. The $\boxed{\text{ADD}}$ button opens a pop-up window into which the necessary resource-attribute values are entered (concluded by pressing the accompanying $\boxed{\text{APPLY}}$ button). The $\boxed{\text{EDIT}}$ button works similarly, opening the aforementioned pop-up window with already populated attribute values that can be edited (and again concluded by pressing the accompanying $\boxed{\text{APPLY}}$ button). (A $\boxed{\text{CANCEL}}$ button is also provided in this pop-up window to abort a creation or editing event.) The $\boxed{\text{DELETE}}$ button deletes the currently selected (i.e., highlighted) resource (note: *without confirmation*). As with all other data-manipulation operations, any creation, edit or deletion may affect the feasible scheduling options for activities elsewhere in the network.

B.4.2.2 Taskforce Manager

The *Taskforce Manager*, depicted in Figure 74, provides a form for editing, modifying and deleting taskforces. As in the *Resource Manager*, the AVAILABLE RESOURCES column on the left side of the form lists all known resources, organized by model type and resource class. The TASKFORCES column on the right lists all known taskforces and their existing constituent resources. New taskforces can be created in the ADD TASKFORCE window in the lower right corner of the form by entering a name and pressing the $\boxed{\text{CREATE}}$ button. Taskforce manipulation is performed using

been received for an activity by the time its start time (either scheduled or not) has passed, it is assumed to be executing within its assigned time window; if an end calltime has not been received for an activity by the time its end time (again, either scheduled or not) has passed, it is assumed to have executed. A more sophisticated model would allow for activities that have not received their expected calltimes to be rescheduled according to the presumably tightened situation. Decisions on how to interpret execution results can be expected to differ by domain and possibly additional environmental factors.


Resource & Taskforce Manager
Available Resources
Selected Resource
3 total capacity
Resource Edit
ADD
EDIT
DELETE
Utilization
1451M
(1D 0H 11M)
Hide resources already assigned to taskforce
Detailed resource information & edit
Taskforce organization
COMIREM
ICLL, Carnegie Mellon University
Activity Network
COMIREM Magboard
Resource & Taskforce Manager
Place Manager
Build Date: October 8, 2003

- MK-V-003
- MK-V-004

fixed-wing
- C-17
  - C-17-001
  - C-17-002
  - C-17-003
  - C-17-004
- AH-64
  - AH-64-001
  - AH-64-002
  - AH-64-003
  - AH-64-004
- AC-130H
  - AC-130H-001
  - AC-130H-002
  - AC-130H-003

ROTOR
- MH-47
  - MH-47-001
  - MH-47-002
  - MH-47-003
  - MH-47-004
  - MH-47-005
- MH-60
  - MH-60-001
  - MH-60-002
  - MH-60-003

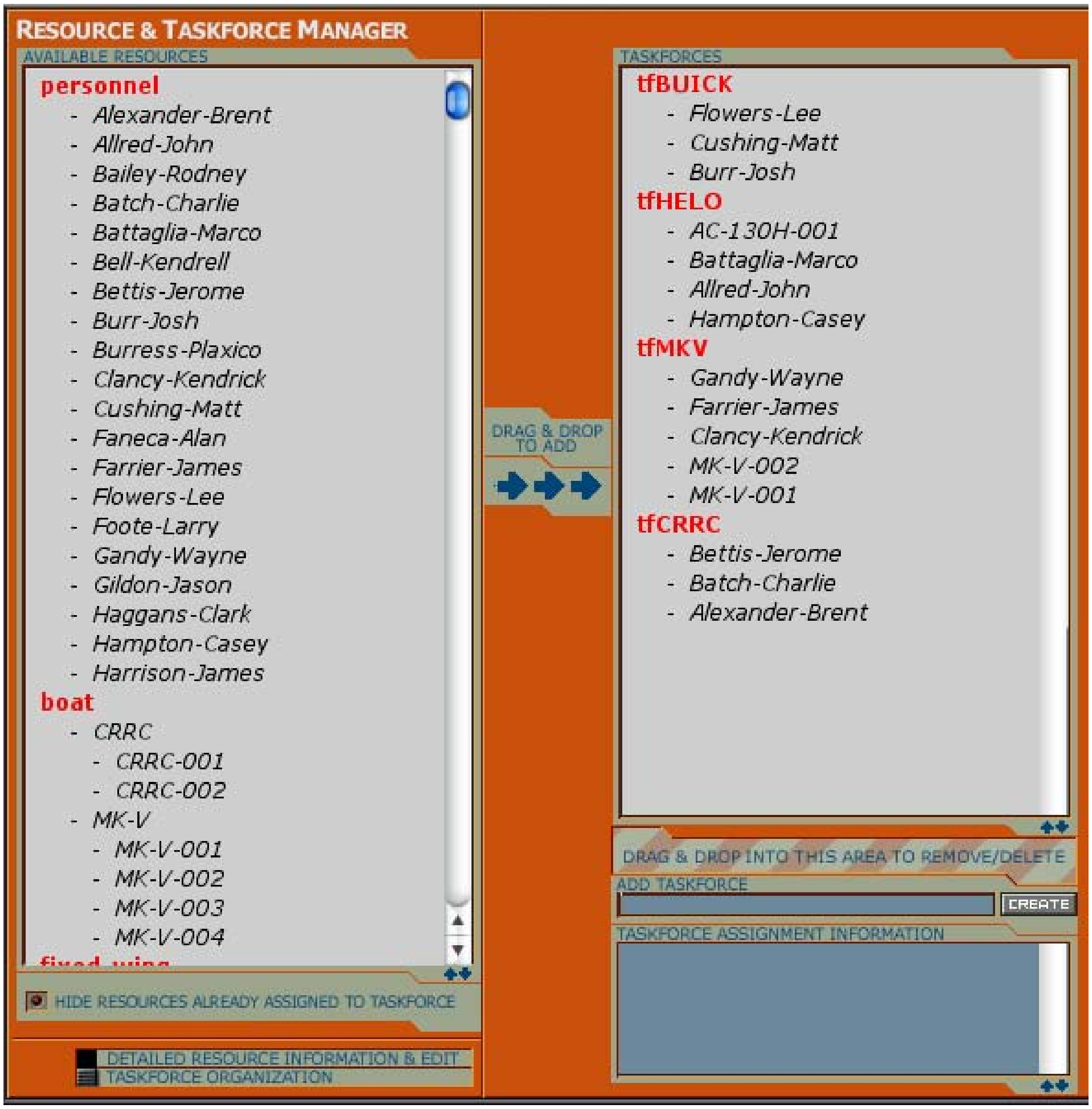
wheeled
- HMMWV
  - HMMWV-001
  - HMMWV-002

Platform	Hull/Tail#	Type	Speed	Place	Group	UIC	Unit	Stat
AC-130H	AC-130H-001	fixed-wing	374	Tampa-Inter	nil	nil	nil	nil
AC-130H	AC-130H-002	fixed-wing	374	FSB-BLACK	nil	nil	nil	nil
AC-130H	AC-130H-003	fixed-wing	374	FSB-BLACK	nil	nil	nil	nil
AH-64	AH-64-001	fixed-wing	374	FSB-BLACK	nil	nil	nil	nil
AH-64	AH-64-002	fixed-wing	374	FSB-BLACK	nil	nil	nil	nil
AH-64	AH-64-003	fixed-wing	374	FSB-BLACK	nil	nil	nil	nil
AH-64	AH-64-004	fixed-wing	374	FSB-BLACK	nil	nil	nil	nil
C-17	C-17-001	fixed-wing	500	Tampa-Inter	nil	nil	nil	nil
C-17	C-17-002	fixed-wing	500	Tampa-Inter	nil	nil	nil	nil
C-17	C-17-003	fixed-wing	500	Tampa-Inter	nil	nil	nil	nil

Figure 73: **Resource & Taskforce Manager** (*Resource Manager Form*)

a drag-and-drop approach. To assign a resource to a taskforce, select it with the mouse and drag it onto the name of the desired taskforce (as depicted in Figure 75). To move a resource from one taskforce to another, select the resource in the TASKFORCES column and drag it onto the name of the desired taskforce (also in the TASKFORCES column). This step is depicted in the left side of Figure 76. To delete a taskforce or remove a resource from a taskforce, select the resource or taskforce from the TASKFORCES column and drag it to the deletion area immediately below the TASKFORCES column. This step (for a taskforce deletion) is depicted in the right side of Figure 76.

The HIDE RESOURCES ALREADY ASSIGNED TO A TASKFORCE radio button located below the AVAILABLE RESOURCES column controls whether resources in the AVAILABLE RESOURCES column are displayed in that column once they have been assigned to a taskforce. While resources can be assigned to multiple taskforces whose time windows do not intersect, it is often the case that resources are assigned to just a single taskforce, in which case the interface can be streamlined by removing those resources once they have been assigned.


RESOURCE & TASKFORCE MANAGER
AVAILABLE RESOURCES
DRAG & DROP TO ADD
TASKFORCES
DRAG & DROP INTO THIS AREA TO REMOVE/DELETE
ADD TASKFORCE
CREATE
TASKFORCE ASSIGNMENT INFORMATION
HIDE RESOURCES ALREADY ASSIGNED TO TASKFORCE
DETAILED RESOURCE INFORMATION & EDIT
TASKFORCE ORGANIZATION

personnel
- Alexander-Brent
- Allred-John
- Bailey-Rodney
- Batch-Charlie
- Battaglia-Marco
- Bell-Kendrell
- Bettis-Jerome
- Burr-Josh
- Burress-Plaxico
- Clancy-Kendrick
- Cushing-Matt
- Faneca-Alan
- Farrier-James
- Flowers-Lee
- Foote-Larry
- Gandy-Wayne
- Gildon-Jason
- Haggans-Clark
- Hampton-Casey
- Harrison-James

boat
- CRRC
  - CRRC-001
  - CRRC-002
- MK-V
  - MK-V-001
  - MK-V-002
  - MK-V-003
  - MK-V-004

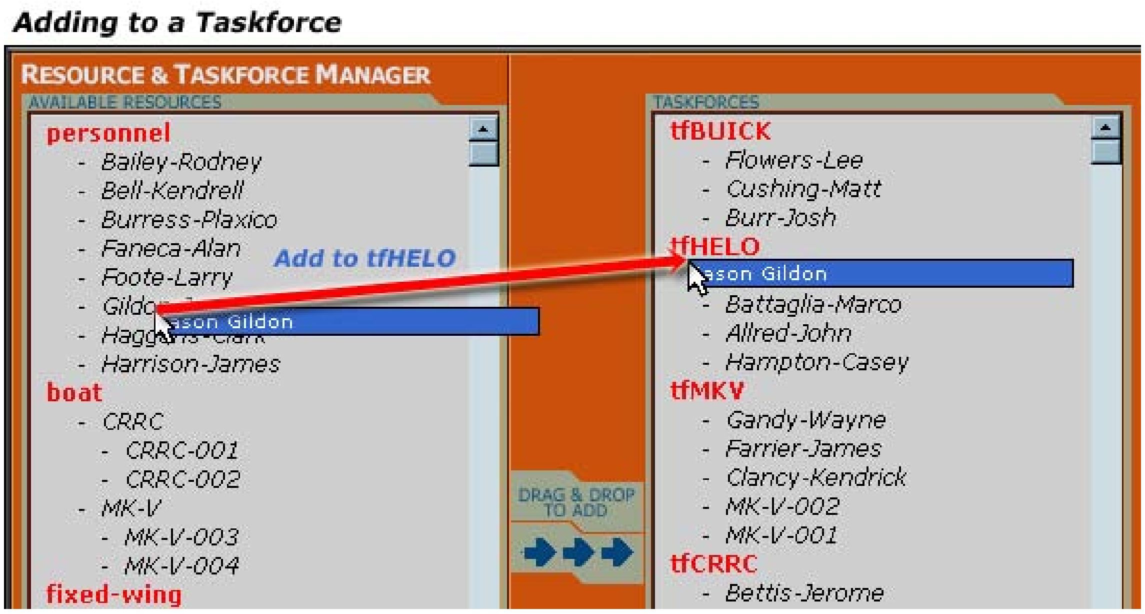
tfBUICK
- Flowers-Lee
- Cushing-Matt
- Burr-Josh

tfHELO
- AC-130H-001
- Battaglia-Marco
- Allred-John
- Hampton-Casey

tfMKV
- Gandy-Wayne
- Farrier-James
- Clancy-Kendrick
- MK-V-002
- MK-V-001

tfCRRC
- Bettis-Jerome
- Batch-Charlie
- Alexander-Brent

Figure 74: **Resource & Taskforce Manager** (*Taskforce Manager Form*)



personnel
- Bailey-Rodney
- Bell-Kendrell
- Burress-Plaxico
- Faneca-Alan
- Foote-Larry
- Gildon-J
- Haggans-Clark
- Harrison-James

boat
- CRRC
  - CRRC-001
  - CRRC-002
- MK-V
  - MK-V-003
  - MK-V-004

fixed-wing

tfBUICK
- Flowers-Lee
- Cushing-Matt
- Burr-Josh

tfHELO
- Jason Gildon
- Battaglia-Marco
- Allred-John
- Hampton-Casey

tfMKV
- Gandy-Wayne
- Farrier-James
- Clancy-Kendrick
- MK-V-002
- MK-V-001

tfCRRC
- Bettis-Jerome

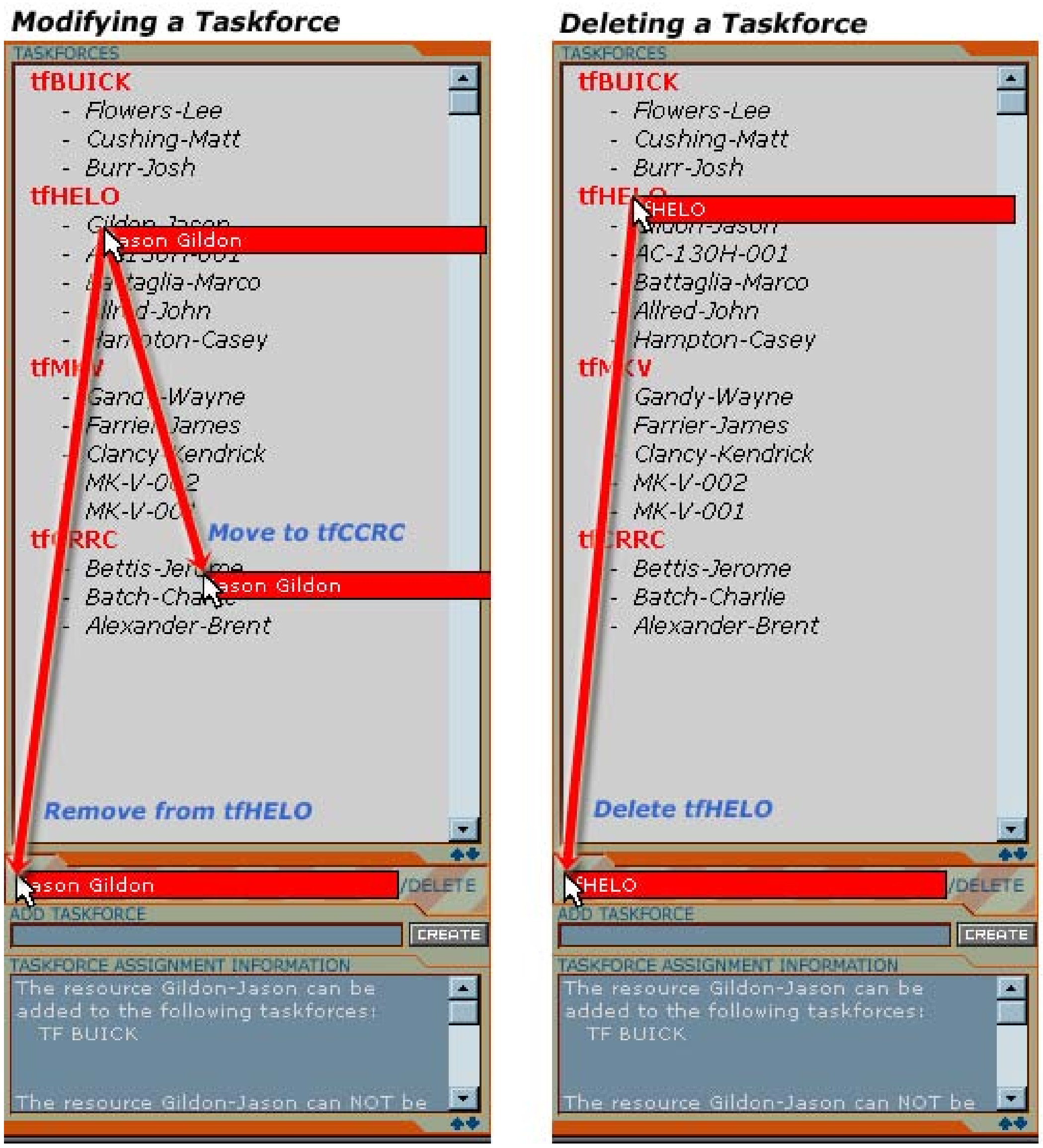
Figure 75: Taskforce management: assigning a (personnel) resource to a taskforce

B.4.3 Place Manager

The **Place Manager**, depicted in Figure 77, provides a form for viewing place capacity and creating, editing and deleting places. Similar to the layout of the **Resource Manager**, this form includes an AVAILABLE PLACES column on the left side that lists all known places. In the lower right corner, the UTILIZATION window displays the capacity of a selected place. In the upper right corner, there are two editing forms: one for platform allocation (the PLATFORM ALLOCATION table), and another for place capacity (the PLACE CAPACITY table).

The PLATFORM ALLOCATION table displays the allocation of platform instances to places, organized by platform-to-place pairings (i.e., there may be multiple rows for each platform and each place in this table). The ADD button opens a new table row into which a known platform name, a known place name and a quantity are entered (concluded by pressing the accompanying APPLY button). The EDIT button works similarly, opening a table row with already populated attribute values that can be edited (and again concluded by pressing the accompanying APPLY button). Again, a CANCEL button aborts either a creation or editing event. The DELETE button deletes the currently selected (i.e., highlighted) platform/place entry (again note: *without confirmation*).

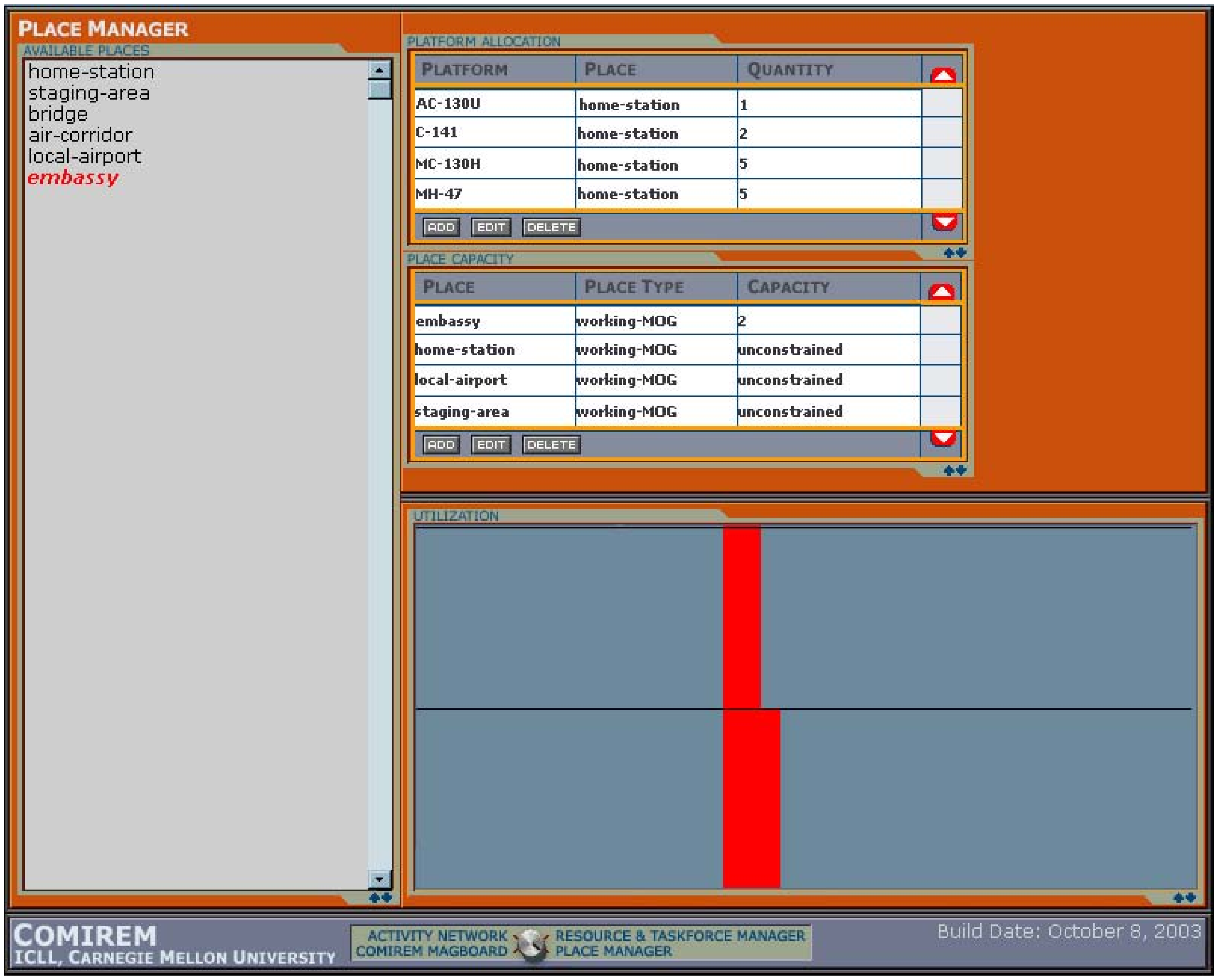
The PLACE CAPACITY table displays place capacity, organized by place-to-component-type pairings (i.e., there may be multiple rows for each place and each component-type in this table). The ADD button opens a new table row into which a known place name, a known component-type name and a quantity are entered (concluded by pressing the accompanying APPLY button). An UNCONSTRAINED checkbox facilitates the specification of unconstrained component-types. The EDIT button works similarly, opening a table row with already populated attribute values that can be edited (and again concluded by pressing the accompanying APPLY button). Again, a CANCEL button aborts either a creation or editing event. The DELETE button deletes the cur-



The resource Gildon-Jason can be added to the following taskforces:
TF BUICK

The resource Gildon-Jason can NOT be

Figure 76: Taskforce management: (left) modifying a taskforce by deleting a member or moving it to another taskforce; (right) deleting a taskforce



home-station
staging-area
bridge
air-corridor
local-airport
embassy

PLATFORM	PLACE	QUANTITY
AC-130U	home-station	1
C-141	home-station	2
MC-130H	home-station	5
MH-47	home-station	5

PLACE	PLACE TYPE	CAPACITY
embassy	working-MOG	2
home-station	working-MOG	unconstrained
local-airport	working-MOG	unconstrained
staging-area	working-MOG	unconstrained

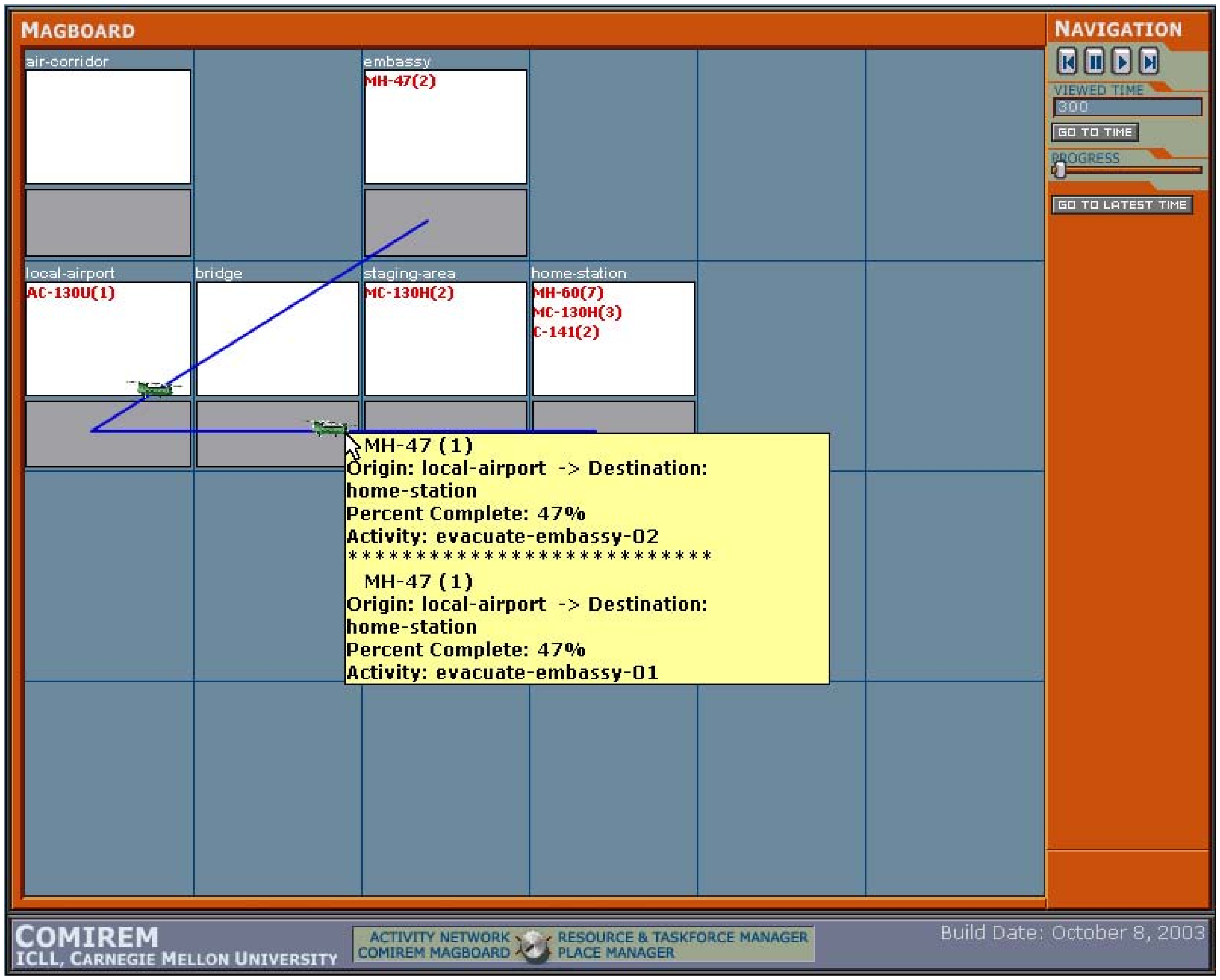
Figure 77: **Place Manager**

rently selected (i.e., highlighted) place/component-type entry (again note: *without confirmation*).

And again, as with all other data-manipulation operations, any creation, edit or deletion may affect the feasible scheduling options for activities elsewhere in the network.

B.4.4 Magboard

The **Magboard**, depicted in Figure 78, is a tool for displaying the location of resources over time. Places are represented by squares on the grid, which is laid out in a simplified geographic format. At any point in time, the white (upper) half of a place square contains a list of all resources currently available at that place, and the gray (lower) half contains a list of all resources currently in-process at that place. For each in-process resource movement between two places, a blue line is drawn with a resource icon placed at its current linearly interpolated position between the two places, according to the schedule. Moving the mouse cursor over a resource icon opens a pop-up window with more information about the movement, including the number of resources involved, its origin and destination, the activity involved, and its completion percentage.



```
 MH-47 (1)
Origin: local-airport  -> Destination:
home-station
Percent Complete: 47%
Activity: evacuate-embassy-02
***************************
 MH-47 (1)
Origin: local-airport  -> Destination:
home-station
Percent Complete: 47%
Activity: evacuate-embassy-01
```

Figure 78: **Magboard**

Initially, the state of the Magboard reflects the beginning of the time horizon (i.e., time 0). The state of the plan(s) at a particular time can be displayed by entering the desired time in the VIEWED TIME field and pressing the GO TO TIME button or by pressing the PLAY icon button. Pressing the PLAY icon button begins a stepping process whereby the VIEWED TIME field is incremented by 10 minutes every 5 seconds and the **Magboard** display is updated to reflect the incremented time until the PAUSE icon button is pressed. To view the location of all resources at the latest calltime, press the GO TO LATEST TIME button. The VIEWED TIME field will be set to the latest calltime and the **Magboard** will display the current state at that time.

C Abstracts of Research Publications

hyperlink ⇝ [Becker and Smith, 2000]

Becker, M.A. and Smith, S.F.
Mixed-Initiative Resource Management: The AMC Barrel Allocator
Proceedings, *Fifth International Conference on Artificial Intelligence Planning Systems (AIPS-00),*
Breckenridge CO, April 2000

Abstract: In this paper, we describe the Barrel Allocator, a scheduling tool developed for day-to-day allocation and management of airlift and tanker resources at the USAF Air Mobility Command (AMC). The system utilizes an incremental and configurable constraint-based search framework to provide a range of automated and semi-automated scheduling capabilities, including generating an initial solution to the fleet assignment problem, selective re-optimization of resource allocations to incorporate new higher priority missions while minimizing solution change, merging of previously planned missions to reduce non-productive flying time, and generation and synchronization of tanker missions to satisfy air refueling requirements. In situations where all mission requirements cannot be met, the system can generate and compare alternative constraint relaxation options. The current version of Barrel Allocator will go into operational use at AMC as a module of Release 2.0 of AMC's Consolidated Air Mobility Planning System (CAMPS) in early 2000.

hyperlink ⇝ [Cesta, Oddi and Smith, 2000b]

Cesta, A., Oddi, A. and Smith, S.F.
Iterative Flattening: A Scalable Method for Solving Multi-Capacity Scheduling Problems
Proceedings, *Seventeenth National Conference on Artificial Intelligence (AAAI-00),*
Austin TX, July 2000

Abstract: One challenge for research in constraint-based scheduling has been to produce scalable solution procedures under fairly general representational assumptions. Quite often, the computational burden of techniques for reasoning about more complex types of temporal and resource capacity constraints places fairly restrictive limits on the size of problems that can be effectively addressed. In this paper, we focus on developing a scalable heuristic procedure to an extended, multi-capacity resource version of the job shop scheduling problem (MCJSSP). Our starting point is a previously developed procedure for generating feasible solutions to more complex, multi-capacity scheduling problems with maximum time lags. Adapting this procedure to exploit the simpler temporal structure of MCJSSP, we are able to produce a quite efficient solution generator. However, the procedure only indirectly attends to MCJSSP's objective criterion and produces sub-optimal solutions. To provide a scalable, optimizing procedure, we propose a simple, local-search procedure called iterative flattening, which utilizes the core solution generator to perform an extended iterative improvement search. Despite its simplicity, experimental analysis shows the iterative improvement search to be quite effective. On a set of reference problems ranging in size from 100 to 900 activities, the iterative flattening procedure efficiently and consistently produces solutions within 10% of computed upper bounds. Overall, the concept of iterative flattening is quite general and provides an interesting new basis for designing more sophisticated local search procedures.

hyperlink ⇝ [Cesta, Oddi and Smith, 2000a]

Cesta, A., Oddi, A. and Smith, S.F.
A Constraint-Based Method For Project Scheduling with Time Windows
***Journal of Heuristics*, vol. 8, no. 1, January 2002, pp. 109–136**

Abstract: This paper presents a heuristic algorithm for solving RCPSP/max, the resource constrained project scheduling problem with generalized precedence relations. The algorithm relies, at its core, on a constraint satisfaction problem solving (CSP) search procedure, which generates a consistent set of activity start times by incrementally removing resource conflicts from an otherwise temporally feasible solution. Key to the effectiveness of the CSP search procedure is its heuristic strategy for conflict selection. A conflict sampling method biased toward selection of minimal conflict sets that involve activities with higher-capacity requests is coupled with a non-deterministic choice heuristic to guide the base conflict resolution process, and this CSP search is embedded within a larger iterative-sampling search framework to broaden search space coverage and promote solution optimization. The efficacy of the overall heuristic algorithm is demonstrated empirically on a large set of previously studied RCPSP/max benchmark problems.

hyperlink ⇝ [Cicirello and Smith, 2001a]

Cicirello, V.A. and Smith, S.F.
Wasp Nests for Self-Configurable Factories
Proceedings, *Fifth International Conference on Autonomous Agents (Agents-01)*,
Montreal Canada, May/June 2001

Abstract: Agent-based approaches to manufacturing scheduling and control have gained increasing attention in recent years. Such approaches are attractive because they offer increased robustness against the unpredictability of factory operations. But the specification of local coordination policies that give rise to efficient global performance and effectively adapt to changing circumstances remains an interesting challenge. In this paper, we introduce a new approach to this coordination problem, drawing on various aspects of a computational model of how wasp colonies coordinate individual activities and allocate tasks to meet the collective needs of the nest. We focus specifically on the problem of configuring machines in a factory to best satisfy (potentially changing) product demands over time. Our system models the set of jobs queued in front of any given machine as a wasp nest, wherein wasp-like agents interact to form a social hierarchy and prioritize the jobs that they represent. Other wasp-like agents external to the nest act as overall machine proxies, and use a model of wasp task allocation behavior to determine which new jobs should be accepted into the machine's queue. We show for simple factories that our multi-agent system achieves the desired effect. For a given job mix, the system converges to a factory configuration that maximizes overall performance, and as the job mix changes, the system quickly adapts to a new, more appropriate configuration.

hyperlink ⇝ [Cicirello and Smith, 2001b]

Cicirello, V.A and Smith, S.F.
Improved Routing Wasps for Distributed Factory Control
Proceedings, *IJCAI-01 Workshop on AI and Manufacturing: New AI Paradigms for Manufacturing*, Seattle WA, August 2001

Abstract: Agent-based approaches to manufacturing scheduling and control are attractive because they offer increased robustness against the unpredictability of factory operations. Previously, we introduced a new approach to coordinating factory routing and scheduling based on a computational model of wasp behavior. The natural multi-agent system of the wasp colony is highly effective in self-organizing the allocation of tasks necessary to fulfill the needs of the nest and has proven useful as an effective model upon which to base our distributed approach to factory control. In this paper, we improve upon our original formulation of the routing wasp with the addition of a tournament of dominance contests among routing wasps competing for the same job. We experimentally evaluate this improved performance.

hyperlink ⇝ [Cicirello and Smith, 2001c]

Cicirello, V.A. and Smith, S.F.
Insect Societies and Manufacturing
Proceedings, *IJCAI-01 Workshop on AI and Manufacturing: New AI Paradigms for Manufacturing*, Seattle WA, August 2001

Abstract: In this paper, we present examples from the literature of successful problem solving systems that have been heavily influenced directly from biological studies of insect societies. Included among these systems are a few of our own. These examples are presented in light of manufacturing applications and only the tip of the iceberg is touched upon. It is our hope that the reader will find a new source of inspiration in problem solving tool design; and realize the potential of coordination mechanisms for multi-agent manufacturing systems inspired by social insects.

hyperlink ⇝ [Cicirello and Smith, 2001d]

Cicirello, V.A. and Smith, S.F.
Randomizing Dispatch Scheduling Policies
Proceedings, *AAAI 2002 Fall Symposium Series: Using Uncertainty within Computation*, North Falmouth MA, November 2001

Abstract: The factory is a complex dynamic environment and scheduling operations for such an environment is a challenging problem. In practice, dispatch scheduling policies are commonly employed, as they offer an efficient and robust solution. However, dispatch scheduling policies are generally myopic, and as such they are susceptible to sub-optimal decision-making. In this paper, we attempt to improve upon results of such dispatch policies by introducing nondeterminism into the decision-making process, and instead using a given policy as a baseline for biasing stochastic decisions. We consider the problem of weighted tardiness scheduling with sequence-dependent setups with unknown arrival times in a dynamic environment, and show that randomization of state-of-the-art dispatch heuristics for this problem in this manner can improve performance. Furthermore, we find that the "easier" the problem, the less benefit there

is from randomization; the "harder" the problem, the more benefit. Our method of randomization is based on a model of the way colonies of wasps self-organize social hierarchies in nature.

hyperlink ⇝ [Cicirello and Smith, 2002a]

Cicirello, V.A and Smith, S.F.
Distributed Coordination of Resources via Wasp-Like Agents
Proceedings, *First International Workshop on Radical Agent Concepts (WRAC-2000)*,
McLean VA, January 2002

Abstract: Agent-based approaches to scheduling have gained increasing attention in recent years. One inherent advantage of agent-based approaches is their tendency for robust behavior; since activity is coordinated via local interaction protocols and decision policies, the system is insensitive to unpredictability in the executing environment. At the same time, such "self-scheduling" systems presume that a coherent global behavior will emerge from the local interactions of individual agents, and realizing this behavior remains a difficult problem. We draw on the adaptive behavior of the natural multi-agent system of the wasp colony as inspiration for decentralized mechanisms for coordinating factory operations. We compare the resulting systems to the state-of-the-art for the problems examined.

hyperlink ⇝ [Cicirello and Smith, 2004]

Cicirello, V.A. and Smith, S.F.
Wasp-like Agents for Distributed Factory Coordination
***Journal of Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, May 2004, pp. 237–266**

Abstract: Agent-based approaches to manufacturing scheduling and control have gained increasing attention in recent years. Such approaches are attractive because they offer increased robustness against the unpredictability of factory operations. But the specification of local coordination policies that give rise to efficient global performance and effectively adapt to changing circumstances remains an interesting challenge. In this paper, we present a new approach to this coordination problem, drawing on various aspects of a computational model of how wasp colonies coordinate individual activities and allocate tasks to meet the collective needs of the nest.

We focus specifically on the problem of configuring parallel multi-purpose machines in a factory to best satisfy product demands over time. Wasp-like computational agents that we call routing wasps act as overall machine proxies. These agents use a model of wasp task allocation behavior, coupled with a model of wasp dominance hierarchy formation, to determine which new jobs should be accepted into the machine's queue. If you view our system from a market-oriented perspective, the policies that the routing wasps independently adapt for their respective machines can be likened to policies for deciding when to bid and when not to bid for arriving jobs.

We benchmark the performance of our system on the real-world problem of assigning trucks to paint booths in a simulated vehicle paintshop. The objective of this problem is to minimize the number of paint color changes accrued by the system, assuming no a priori knowledge of the color sequence or color distribution of trucks arriving in the system. We demonstrate that our system outperforms the bidding mechanism originally implemented for the problem as well as another related adaptive bidding mechanism.

hyperlink ⇝ [Cicirello and Smith, 2002b]

Cicirello V.A. and Smith, S.F.
Amplification of Search Performance through Randomization of Heuristics
Proceedings, *Eighth International Conference on Principles and Practice of Constraint Programming,*
Ithaca NY, September 2002

Abstract: Randomization as a means for improving search performance in combinatorial domains has received increasing interest in recent years. In optimization contexts, it can provide a means for overcoming the deficiencies of available search heuristics and broadening search in productive directions. In this paper, we consider the issue of amplifying the performance of a search heuristic through randomization. We introduce a general framework for embedding a base heuristic within an iterative sampling process and searching a stochastic neighborhood of the heuristic's prescribed trajectory. In contrast to previous approaches, which have used rank-ordering as a basis for randomization, our approach instead relies on assigned heuristic value. Use of heuristic value is important because it makes it possible to vary the level of stochasticity in relation to the discriminatory power of the heuristic in different decision contexts, and hence concentrate search around those decisions where the heuristic is least informative. To evaluate the efficacy of the approach, we apply it to a complex, weighted-tardiness scheduling problem. Taking a state-of-the-art heuristic for this scheduling problem as a starting point, we demonstrate an ability to consistently and significantly improve on the deterministic heuristic solution across a broad range of problem instances. Our approach is also shown to consistently outperform a previously developed, rank-ordering based approach to randomizing the same heuristic in terms of percentage of improvement obtained.

hyperlink ⇝ [Cicirello, 2003]

Cicirello, V.A.
Boosting Stochastic Problem Solvers Through Online Self-Analysis of Performance
Ph.D. Thesis, Technical Report CMU-RI-TR-03-27,
The Robotics Institute, Carnegie Mellon University, July 2003

Abstract: In many combinatorial domains, simple stochastic algorithms often exhibit superior performance when compared to highly customized approaches. Many of these simple algorithms outperform more sophisticated approaches on difficult benchmark problems; and often lead to better solutions as the algorithms are taken out of the world of benchmarks and into the real-world. Simple stochastic algorithms are often robust, scalable problem solvers.

This thesis explores methods for combining sets of heuristics within a single stochastic search. The ability of stochastic search to amplify heuristics is often a key factor in its success. Heuristics are not, however, infallible and in most domains no single heuristic dominates. It is therefore desirable to gain the collective power of a set of heuristics; and to design a search control framework capable of producing a hybrid algorithm from component heuristics with the ability to customize itself to a given problem instance. A primary goal is to explore what can be learned from quality distributions of iterative stochastic search in combinatorial optimization domains; and to exploit models of quality distributions to enhance the performance of stochastic problem solvers. We hypothesize that models of solution quality can lead to effective search control mechanisms, providing a general framework for combining multiple heuristics into an enhanced decision-making process. These goals lead to the development of a search control framework, called QD-BEACON, that uses online-generated statistical models of search performance to effectively combine search heuristics. A prerequisite goal is to develop a suitable stochastic sampling algorithm for combinatorial search problems. This goal leads to the development of an algorithm called VBSS that makes better use, in general, of the discriminatory power of a given search heuristic as compared to existing sampling approaches.

The search frameworks of this thesis are evaluated on combinatorial optimization problems. Specifically, we show that: 1) VBSS is an effective method for amplifying heuristic performance for the weighted tardiness sequencing problem with sequence-dependent setups; 2) QD-BEACON can enhance the current best known algorithm for weighted tardiness sequencing; and 3) QD-BEACON and VBSS together provide the new best heuristic algorithm for the constrained optimization problem known as RCPSP/max.

hyperlink ⇝ [Derthick and Smith, 2004]

Derthick, M. and Smith, S.F.
An Interactive 1D+2D+3D Visualization for Requirements Analysis
***Journal of Scheduling* 2004 (to appear)**

Abstract: In most practical domains, scheduling is not a one-shot generative task of producing time and resource assignments for pre-specified sets of requirements and capabilities, but an iterative process of getting the constraints right. Initial solutions are developed (at some level of detail) to understand the feasibility of various requirements and the sufficiency of assumed resources. This requirements analysis leads to reassessment of what requirements can be reasonably met and what resources need to be made available. Changes are made to the constraints governing requirements and resource availability, and eventually a final schedule is elaborated. At early stages of this user-driven process, seeing a fully instantiated schedule is less important than simply knowing whether a feasible schedule exists and, if not, the approximate magnitude of the shortfalls for different resources and periods of time. To this end, solutions to relaxed versions of the full scheduling problem can provide helpful guidance.

In this paper, we describe a system that builds on this notion to provide a direct-manipulation visual interface for requirements analysis and reconciliation. The interface incorporates a 3D visualization that identifies resource capacity shortfalls in a tractable relaxed version of the problem, which must necessarily also be shortfalls in the original problem. Our visualization can be contrasted with common 2D scheduling displays such as Gantt Charts and Closure Graphs, which are designed for visualizing complete solutions to a given scheduling problem and hence offer only indirect support for identifying inherently over-constrained regions of the solution space. Our visualization, alternatively, directly characterizes this underlying constraint space and provides a direct basis for requirements analysis. An analyst iteratively adjusts problem constraints and visualizes the resulting (relaxed) problem solution until various mismatches between resource requirements and resource availability have been satisfactorily reconciled. Once a reasonable compromise has been found, the same interface can then be used to guide more detailed scheduling.

hyperlink ⇝ [Kramer and Smith, 2003]

Kramer, L.A. and Smith, S.F.
Maximizing Flexibility: A Retraction Heuristic for Oversubscribed Scheduling Problems
Proceedings, *Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*,
Acapulco Mexico, August 2003

Abstract: In this paper we consider the solution of scheduling problems that are inherently over-subscribed. In such problems, there are always more tasks to execute within a given time frame than available resource capacity will allow, and hence decisions must be made about which tasks should be included in the schedule and which should be excluded. We adopt a controlled, iterative repair search approach, and focus on improving the results of an initial priority-driven solution generation procedure. Central to our approach is a new retraction heuristic, termed max-flexibility, which is responsible for identifying which tasks to (temporarily) retract from the schedule for reassignment in an effort to incorporate additional tasks into the schedule. The max-flexibility heuristic chooses those tasks that have maximum flexibility for assignment within their feasible windows. We empirically evaluate the performance of max-flexibility using problem data and the basic scheduling procedure from a fielded airlift mission scheduling application. We show that it produces better improvement results than two contention-based retraction heuristics, including a variant of min-conflicts [Minton et.al] with significantly less search and computational cost.

hyperlink ⇝ [Nareyek, Smith and Ohler, 2003]

Nareyek, A., Smith, S.F. and Ohler, C.M.
Integrating Local Search Advice into Refinement Search (Or Not)
Proceedings, *CP-03 Third Workshop on Cooperative Solvers in Constraint Programming (COSOLV 2003)*,
Kinsale, County Cork, Ireland, September 2003

Abstract: Recent work has shown the promise in using local-search "probes" as a basis for directing a backtracking-based refinement search. In this approach, the decision about the next refinement step is based on an interposed phase of constructing a complete (but not necessarily feasible) variable assignment. This assignment is then used to decide on which refinement to take, i.e., as a kind of variable- and value-ordering strategy.

In this paper, we further investigate this hybrid search approach. First, we evaluate methods for improving probe-based guidance, by basing refinement decisions not only on the final assignment of the probe-construction phase but also on information gathered during the probe-construction process. Second, we consider the relative strengths of probe-based search control and search control that is biased by more classically motivated variable- and value-ordering heuristics (incorporating domain-specific knowledge). The approaches are evaluated on various problems from the job-shop scheduling domain.

Our results indicate that — while probe-based search performs better than an uninformed search — use of domain-specific knowledge proves to be a much more effective basis for search control than information about constraint interactions that is gained by local-search probes, and leads to substantially better performance.

hyperlink ⇝ [Smith, 2003]

Smith, S.F.
Is Scheduling a Solved Problem?
Proceedings, *First Multi-Disciplinary International Conference on Scheduling: Theory and Applications (MISTA)*,
Nottingham UK, September 2003

(An earlier version of this paper appeared in Proceedings, *GECCO 2001 Workshop on the Next Ten Years of Scheduling Research*, San Francisco CA, July 2001.)

Abstract: In recent years, scheduling research has had an increasing impact on practical problems, and a range of scheduling techniques have made their way into real-world application development. Constraint-based models now couple rich representational flexibility with highly scalable constraint management and search procedures. Similarly, mathematical programming tools are now capable of addressing problems of unprecedented scale, and meta-heuristics provide robust capabilities for schedule optimization. With these mounting successes and advances, it might be tempting to conclude that the chief technical hurdles underlying the scheduling problem have been overcome. However, such a conclusion (at best) presumes a rather narrow and specialized interpretation of scheduling, and (at worst) ignores much of the process and broader context of scheduling in most practical environments. In this note, I argue against this conclusion and outline several outstanding challenges for scheduling research.

hyperlink ⇝ [Smith, Becker and Kramer, 2004]

Smith, S.F., Becker, M.A. and Kramer, L.A.
Continuous Management of Airlift and Tanker Resources: A Constraint-based Approach
***Mathematical and Computer Modelling*,**
Special Issue on Defense Transportation:
Algorithms, models, and Applications for the 21st Century,
vol. 39, nos. 6–8, March 2004, pp. 581–598

Abstract: Efficient allocation of aircraft and aircrews to transportation missions is an important priority at the USAF Air Mobility Command (AMC), where airlift demand must increasingly be met with less capacity and at lower cost. In addition to presenting a formidable optimization problem, the AMC resource management problem is complicated by the fact that it is situated in a continuously executing environment. Mission requests are received (and must be acted upon) incrementally, and, once allocation decisions have been communicated to the executing agents, subsequent opportunities for optimizing resource usage must be balanced against the cost of solution change. In this paper, we describe the technical approach taken to this problem in the AMC Barrel Allocator, a scheduling tool developed to address this problem and provide support for day-to-day allocation and management of AMC resources. The system utilizes incremental and configurable constraint-based search procedures to provide a range of automated and semi-automated scheduling capabilities. Most basically, the system provides an efficient solution to the fleet scheduling problem. More importantly to continuous operations, it also provides techniques for selectively re-optimizing to accommodate higher priority missions while minimizing disruption to most previously scheduled missions, and for selectively "merging" previously planned missions to minimize non-productive flying time. In situations where all mission requirements cannot be met, the system can generate and compare alterna-

tive constraint relaxation options. The Barrel Allocator technology is currently transitioning into operational use within AMC's Tanker/Airlift Control Center (TACC). A version of the Barrel Allocator supporting airlift allocation was first incorporated as an experimental module of the AMC's Consolidated Air Mobility Planning System (CAMPS) in September 2000. In May 2003, a new tanker allocation module is scheduled for initial operational release to users as part of CAMPS Release 5.4.

hyperlink ⇝ `[Smith, Hildum and Crimm, 2003]`

Smith, S.F., Hildum, D.W. and Crimm, D.R.
Interactive Resource Management in the COMIREM Planner
Proceedings, *IJCAI-03 Workshop on Mixed-Initiative Intelligent Systems*,
Acapulco Mexico, August 2003

Abstract: In this paper, we describe Comirem, a light-weight, interactive tool for resource management in continuous planning domains. Comirem is designed for domains where complex, heterogeneous sets of resources are required to execute planned activities and usage must be synchronized to satisfy complex temporal and spatial constraints. Comirem promotes an opportunistic planning paradigm, where resource allocation decisions are made and revised incrementally as plans and availability constraints become known and refined during the planning process, and as execution deviates from planned behavior. To this end, resources are assigned (and re-assigned) via a least-commitment, constraint-posting scheduling procedure. The Comirem system design follows three basic mixed-initiative principles: (1) that users will want to make planning and resource allocation decisions at different levels of detail in different circumstances and correspondingly delegate more or less decision-making responsibility to system processes in different contexts, (2) that abstract domain models, coupled with graphical visualization can provide an effective basis for communicating decision impact and proposing decision options, and (3) that incremental, adaptive problem-solving capabilities, which attempt to localize change whenever possible and appropriate, are central to maintaining continuity in the planning and resource management process. These principles are used to provide a variety of tools for mixed-initiative resource allocation, feasibility checking, resource tracking and conflict resolution.

hyperlink ⇝ `[Smith, Hildum and Crimm, 2001]`

Smith, S.F., Hildum, D.W. and Crimm, D.R.
Toward the Design of Web-Based Planning and Scheduling Services
Proceedings, *ECP-01/PLANET Workshop on Automated Planning and Scheduling Technologies in New Methods of Electronic, Mobile and Collaborative Work*,
Toledo Spain, September 2001

Abstract: This paper introduces COMIREM (Continuous, Mixed-Initiative Resource Management), a system for collaborative, incremental development of plans and schedules in dynamic, resource-constrained domains. COMIREM is designed to provide web-based planning and scheduling services and is capable of interacting with a variety of interfacing and visualization tools using a standard Internet browser. The objective is to deliver a wide range of mixed-initiative problem-solving capabilities (e.g., specification of activities and requirements, commitment/de-commitment of resources, manipulation of problem constraints) through light-

weight plug-in applications that can be selected according to present circumstances. We present a summary of the current functionality and an outline of the developing system architecture.

hyperlink ⇝ [Zhou and Smith, 2002a]

Zhou, Q. and Smith, S.F.
A Priority-based Preemption Algorithm for Incremental Scheduling with Cumulative Resources
Technical Report CMU-RI-TR-02-19,
The Robotics Institute, Carnegie Mellon University, July 2002

Abstract: When scheduling in dynamic continuous environments, it is necessary to integrate new tasks in a manner that reflects their intrinsic importance while at the same time minimizing solution change. A scheduling strategy that re-computes a schedule from scratch each time a new task arrives will tend to be quite disruptive. Alternatively, a purely non-disruptive scheduling strategy favors tasks that are already in the schedule over new ones, regardless of respective priorities. In this paper, we consider algorithms that attempt to strike a middle ground. Like a basic non-disruptive strategy, the algorithm we propose emphasizes incremental extension/revision of an existing schedule, rather than regeneration of a new schedule from scratch. However, by allowing selective preemption of currently scheduled tasks, our algorithm also gives attention to the relative importance of new tasks. We consider a specific class of scheduling problems involving the allocation of cumulative (or multi-capacity) resources. We develop an approach to preemption based on the concept of freeing up a resource area (i.e., time and capacity rectangle) comparable to the resource requirement of the new task to be scheduled. Through experimental analysis performed with a previously developed system for air combat operations scheduling, we demonstrate that our priority-based preemption algorithm is capable of producing results comparable in solution quality to those obtained by regenerating a new schedule from scratch with significantly less disruption to the current schedule.

Zhou, Q. and Smith, S.F.
An Efficient Consumable Resource Representation for Scheduling
Proceedings, *Third International NASA Workshop on Planning and Scheduling for Space*,
Houston TX, October 2002

Abstract: This paper proposes a new capacity representation for consumable resources, which uses an expanded balanced tree as its data structure. Its efficiency is analyzed and demonstrated by empirical experiments. Algorithms for the core operations required of a scheduling system (i.e. check available capacity, allocate capacity and de-allocate capacity) are also provided and proved to take $O(\mathit{log}\ n)$ time in the worst case.

□